Applications of solid modelling to component inspection with coordinate measuring machines

Walker, Ian

Applications of Solid Modelling To Component Inspection with Coordinate Measuring Machines

Submitted by

Ian Walker

for the degree of Doctor of Philosophy

of the University of Bath

1991

[illegible]

OVERVIEW

Inspection by coordinate measuring machines (CMMs) is widely used to assess the quality of manufactured components. One purpose of such inspection is to estimate the accuracy of machined geometric features and check that they are manufactured within specified tolerances. CMM inspection is slow, and time could be saved by automating the planning and measurement analysis stages of the inspection process.

Solid modelling is a means of representing three dimensional solid objects using computers complete with geometric information, and has a wide range of engineering applications. Because the models contain geometric information, solid modelling is also useful for automating CMM inspection.

The work described in this thesis involves using set theoretic solid modelling to automate stages of the CMM inspection process and may be split into three parts:

(i) Developing a method of attaching geometrical tolerances to solid models.

(ii) Using the toleranced model to generate an inspection plan, consisting of points to measure and a probe path.

(iii) Using the results of inspection and the model to verify that the measured features lie within the given tolerances.

Tolerances are attached as a form of attribute to the solid model and are added as the model is defined. Such attributes allow other non-geometric information to be assigned to the model, and thus enable, for example, control parameters for inspection planning to be defined.

The inspection planning involves first automatically generating inspection points using appropriate inspection attributes and the tolerance information. Then the solid model is used to determine a path through the points for the CMM measuring probe. This path is collision-free and also is constructed with the aim of minimising the time spent on inspection.

Finally the measured results are collected and used with the toleranced solid model to check that the various geometric tolerances have not been violated. The checking is done as defined in the British Standards BS308 (Part 3 1972) on geometric tolerances.

ACKNOWLEDGEMENTS

I would like to thank Dr Andrew Wallis for his valuable input and support to the work described within this thesis.

In addition I would like to thank Dr Adrian Bowyer, Dr Zhirong Li and Dr Yun-feng Zhang for their assistance during the writing of this thesis.

Finally I would like to thank all others who have contributed to this work.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter describes inspection by Coordinate Measuring Machines (CMM) and current working practices. The theory of solid modelling is then described and this is followed by a description of the areas where the inspection process can be automated. The aims of the project described in this thesis are then given followed by an overview of the resulting automated inspection system.

The chapter concludes with a description of the layout of the thesis.

1.2 Inspection of Machined Components

Once engineering components have been designed they are manufactured using various machining and forming techniques. An important subset of the range of manufacturing techniques is that of metal-cutting or machining, and the work described in this thesis is concerned with machined components.

It is necessary to have some means of checking whether the resulting manufactured component fulfills the original design specification, and this is done by the process of *inspection*. Component inspection enables the quality of production to be monitored, and also provides useful feedback into the manufacturing process. Such feedback can be used to correct errors or to detect machining faults.

Inspection is a means of collecting information about the shape and dimensions of a machined component using various measuring procedures. The results of such measurements are then used to assess any discrepancies between the original specification and the final component. Constraints are applied to specify the acceptable deviation of the machined component from the original design using *tolerances.* The nature of such constraints will depend on the application for which the component is being developed.

Two types of tolerances can be applied to a component;

(i) Dimensional Tolerances - These constrain the allowed variation of dimensions (e.g such as length, radii)

(ii) Geometrical Tolerances - These constrain the allowed variation of geometrical properties (e.g the Squareness of one plane to another, the Roundness of a cylinder)

British Standards BS308 defines the standards used for representation and interpretation of both tolerances and should be consulted for further information. Tolerances are currently represented within the 2-Dimensional Engineering Drawings of the component.

The component specification (e.g. engineering drawings) will define its dimensions and required shape. Inspection for checking the shape involves measuring geometrical dimensions of the component. There are various ways of collecting such information:

(a) Manually using various general purpose measuring instruments, each designed to measure a particular geometric dimension or angle.

(b) Automatically using computer-controlled CMM which use touch-sensitive probes to measure the component shape.

(c) Automatically using computer vision inspection systems which use cameras, and image analysis packages to check the accuracy of component shape.

Collecting information manually is slow and the growing use of numerically controlled machine-tools has meant that methods of faster inspection are needed to keep up with production. Added to this are the problems of human error during inspection which may result in inaccurate or inconsistent results. In view of this, techniques of automating inspection have been developed.

Automatic inspection of component geometry has been done using touch-sensitive measuring probes. These probes indicate when contact is made with the surface of an object, and together with suitable feedback devices for determining the probe position, are used to collect the three dimensional coordinates of points on the surface of the component.

The probe may, in principle, either be incorporated into the machine-tools, or be external to the machining environment in numerically controlled Coordinate Measuring Machines. The concept of inspection as part of the machine cycle appears attractive, with immediate feedback being available to adaptively control the machining process. However, current technology does not enable measurements to be made with the accuracy required for the inspection of geometric shape. The harsh cutting environment hinders such accuracy, where effects due to temperature and clamping of the component may be considerable. Further, the computing power of an average machine-tool adds constraints to the methods of analysis of measurement results, and only allows limited feedback control.

As a contrast, CMM inspection allows accuracy within the limits of the measuring equipment and as a result has become readily accepted as part of the quality control aspect of the manufacturing process. Further, should technology improvements enable accurate inspections during cutting, techniques developed for CMMs could be easily adapted.

Automated computer vision inspection offers an alternative to CMM inspection. However, once again current technology does not enable measurement to the required accuracy for inspection of geometric shape, and most visual inspection systems are accompanied by CMM technology for critical measurements.

1.3 Coordinate Measuring Machines

The British Standard BS6808 defines the coordinate measuring machine as:

"A machine having a series of movable members, a sensing probe and a workpiece support member which can be brought into a fixed and known relationship with points on the workpiece surface, and the coordinates of these points can be displayed or otherwise determined with respect to the origin of the coordinate system".

The same standard also describes the variety of different types of CMMs, which are used to collect three-dimensional information about a test component in the form of three-dimensional coordinates of surface points.

The measuring machines are used in manufacturing processes which rely heavily on precision engineering. Rolls-Royce, Lister-Petter and Bendix Ltd are examples of companies who use CMMs for inspection, with the machines installed on the shop floor. CMMs are used in applications covering a broad range of manufacturing batch sizes from total inspection of small batches to sample inspection of large batches.

The machines are generally connected to a small computer which drives the probe around the test component probing the surface at pre-determined locations. In some instances, the computer also does some simple analysis of the measured results and displays the feature information resulting, for example, from least-square fits of surfaces to the measured points. Such information is used to compare the nominal with the measured geometry.

In current practice, the selection of points and the choice of path is done manually and prior to the inspection. This requires a certain amount of skill in deciding which location points to inspect and in deciding how many points should be inspected. The choice of point location depends on the geometry and application of the various component features which are being measured. The number of points depends on the size of the features and the types of tolerances applied to them. Both the location and the number of points are influenced by the type of analysis carried out on the results to check the geometric shape.

Methods of constructing inspection paths for CMMs vary, but fall into two categories. One technique is for technicians to "teach" the path manually by moving the CMM through the motions that constitute the inspection path. The sequence of path steps saved to a file may be replayed by CMMs used to inspect components. This technique is used at Rolls-Royce and Lister-Petter and, although being adaptable to various components and probe types, clearly is tedious and time consuming. A second technique is to produce inspection paths using off-line part-programming in much the same way that numerically controlled machine-tool paths are generated. Bendix, for example, adopts this approach which, although having some advantages over manual programming, is still time consuming and also needs verification tests before use.

When replaying the taught or programmed path, the CMMs controller interprets the path instructions and drives the probe to collect the measurements. The paths usually includes several datum points which are used to align the coordinate system of the component and the CMM. By making the assumption that the orientation of the component is roughly as expected and by measuring the datum points, a transformation between the two coordinate systems is derived. This transformation is used to modify the inspection locations defined in the probe path to obtain the correct points on the component. This removes the need for jigs and clamps for holding the component in a fixed position.

In many cases only limited analysis of the results of measuring are performed. Currently, comparisons between original and measured parameters are the main output of CMM inspection.

1.4 Solid Modelling: Description

In order to automate the inspection planning and measurement analysis stages of CMM inspection, a computer representation of the geometry of the component is required. *Solid modelling* is a technique of representing the geometry of three-dimensional objects on a computer and the work in this thesis is concerned with the application of such modelling to the automation CMM inspection.

Methods and techniques of solid modelling vary, but all involve building the model by enclosing the volume occupied by the component with surfaces. Effectively, three-dimensional space is divided into two regions, one representing the object solid, the other representing air. A solid modelling system usually comprises a data structure containing the geometric description of objects together with a set of procedures for performing various tasks, such as the calculation of the volume or the generation of pictures of the model.

The solid modelling scheme must fulfill several requirements to enable representation of real objects. The representation must be rigid, be invariant under geometric transformations, and also must only represent physically realisable shapes. Other desirable properties of the modelling system include ease of use and the ability to support efficient and numerically robust algorithms for computation tasks.

There are many solid modellers in current use, and most can be classified into one of two types: boundary representations, or set theoretic representations. The difference, as described in section 1.5, lies in the information used to represent the object and both techniques of solid modelling have advantages and disadvantages. Some solid model systems use a hybrid approach which exploits the advantages of both representations but also inherits some of the disadvantages.

1.5 Boundary Representation and Set Theoretic Modelling

The approach most commonly used in commercial solid modelling systems is boundary representation. Boundary representation stores models of objects as a list of edges, vertices and surfaces. A complicated data structure is used to relate these entities in such a way as to define the object. Consistency is generally checked using rules or formulae. As an example, Eulers' formulae, which define relationships between the number of faces, edges and surfaces, are often used in a valid boundary representation model to check consistency of shape.

In boundary representations, certain geometrical information (e.g. vertices, edges) is readily available and little or no processing is required to extract it. Further, the representations are generally unambiguous, being explicitly defined by vertices, edges and surfaces.

The problems of such a representation are well known and described in Section 2.4.1. In summary, apart from requiring the complicated data structure and definition procedures, it is also difficult to ensure the representation is valid and consistent.

Set theoretic solid modellers do not exhibit this problem. In this scheme, an object is represented as the set theoretic combination of simpler objects, known as *primitives*. The set theoretic operators union, difference and intersection are used to build the required shape from these primitives. The primitives can be a finite set of simple bounded shapes, such as cylinders, cuboids and spheres, or can be semi-infinite surfaces, known as a *half spaces*.

A half space surface splits three-dimensional space into two regions; one region consisting of points above the surface and one region consisting of points below it. A solid model can be constructed with half spaces by using one of the two regions to represent object solid and by using the other region to represent the air outside the object solid.

Set theoretic solid modelling guarantees consistency of representation, and thus always represents a physically realisable object. The data structures needed to store such models are simple to build, with no concern for vertices, edges and surfaces or how they influence each other. Drawbacks are non-uniqueness of representation and the difficulty of extraction of geometric information. The latter arises because, in general, the geometric information of the component cannot always be derived *directly* from the information on the constituent primitives since not all of a primitive surface is guaranteed to lie on the model surface. The primitive contribution to the surface, for example, may be effected by another, separate primitive.

Interactive modification is an expensive process in both modelling schemes. Boundary representations require recalculation of edges and vertices, and changes to set theoretic models require the recalculation of the complete set theoretic expression. The boundary scheme does have the advantage of being able to easily support local changes since it has all the geometric information explicitly available and this can be used to isolate areas where recalculation is required. Such localisation is not possible with set theoretic models where it is difficult to predict how changes to the primitives within the model will effect the complete model. This has been a factor in encouraging the commercial applications of boundary representations, where the major use seems to be in interactively building pictures of components.

Current hybrid solid modellers usually allow models to be built as set theoretic expressions and then generate a boundary representation for subsequent processing. It is a much more difficult problem, however, to constuct a set theoretic model from a boundary representation.

The work in this thesis has been developed for a set theoretic modelling scheme in order to take advantage of the properties of robustness, consistency and simplicity of construction. As mentioned above, boundary representations can (and are) be generated from set theoretic models although the process is computationally expensive and should only be done as and when it is necessary.

1.6 Automating CMM Inspection

There are two obvious candidates for automation during the various stages of CMM inspection; automation of inspection planning and automation of the anaylsis of the measurements collected by the CMM.

1.6.1 Automation of Inspection Planning

CMMs were introduced to reduce inspection time without affecting accuracy and also to enable integration of inspection into Flexible Manufacturing Systems. Further integration of CMM inspection is possible by automating the inspection path-planning stages. This will provide a number of benefits including:

(a) More efficient inspection systems as human interaction would no longer be required. The manual planning of inspection relies on human interpretation of engineering drawings, and this in turn is dependent on the experience and skill of the technicians. Automatic planning would remove this dependency.

(b) Provision of a basis for standardising current industrial inspection practices and thus encouraging a consistently high level of manufacturing quality.

(c) The time taken for inspection planning is never a bottleneck to the manufacturing process, but it is probably still worth mentioning that by automating the planning of inspection the speed of the task is made dependent on the always improving processing power of computers.

1.6.2 Automatic Analysis of Measurements

There are three steps in the process of automating the analysis of measurements made by CMM using a solid modelling system;

(a) Representation of Tolerances

A scheme is required for representing tolerances within the solid model. This is a non-trivial problem for both types of tolerances (and both Set theoretic and Boundary Rep Models). Dimensional tolerances require references to dimensions within the model and geometric tolerances require references to geometric entities within the model, neither of which are available within solid models. Extra non-geometric information is required (such as labels of geometric entities) to support such references.

(b) Validation of Toleranced Models

Tolerances (dimensional and geometrical) applied to components do interact and it is possible to apply tolerances in such a way as to make manufacturing the component impossible (e.g. keeping to one tolerance may always violate another). Once tolerances have been applied to a component, the toleranced model should be validated to ensure that such inconsistencies have not occurred. The problem is a formidable task to do manually but can be automated since interaction of engineering tolerances is a well understood area. Since, however, it is a problem concerned with the nature of engineering tolerances rather than component inspection, it is considered to be beyond the scope of this thesis.

(c) Interpretation and Checking of Tolerances

The dimensional and geometrical tolerances applied to the model have to be interpreted in terms of constraints which can be measured by the CMM. The measurements collected by CMM are then used to ascertain whether the machined component fulfills the design requirements and check that none of the tolerances specified in the design are violated. By having computer access to geometric and tolerance information of the component, rigorous analysis is possible, thus allowing automatic detection of tolerance violations.

The work in this thesis has restricted attention to the problem of representation and interpretation of geometric tolerances within set theoretic models. This is because of the following reasons;

(a) Set theoretic models are built constructively by combining geometric primitives. Thus the geometric entities are defined independently before being combined to construct the object. This offers a means of representing geometric tolerances as attributes of geometric entities and enables their specification at the same time the entities are defined. The need for a complicated referencing scheme is thus reduced, although some means of referencing datum entities is still required.

(b) The interpretation of geometric tolerances is in terms of 3-Dimensional constraints which can in many cases be checked using CMM measurements directly. Dimensional tolerances are defined within a 2-Dimensional context (i.e. within the view of an Engineering Drawing) and require further processing in order to determine the constraints which they apply within 3-Dimensions. (This processing can be quite daunting especially when the constraints on edges and vertices are concerned).

(c) Many of the constraints imposed by dimensional tolerances can be represented as geometrical tolerances.

(d) Consideration of geometric tolerances and solving the problems of their representation and interpretation within solid models provides a foundation for research into the problems posed by dimensional tolerances.

1.7 Objectives of This Research

The main objective of this research is to automate the planning stage and the analysis of measurement results stage of component inspection by Coordinate Measuring Machines. One approach to inspection planning would be to automatically generate the inspection plan from geometric information of the component stored on a computer. This approach automates the stages where technicians currently do the work manually. The objective of the work described within this thesis is to use set theoretic solid model representation of components as a source of geometric information for automating CMM inspection.

The models are constructed complete with geometric tolerance information. This information is used in making decisions about inspection locations on the component surface and also is used to define the necessary tests which are needed on the resulting measurement data during the analysis stage.

Few current solid modellers allow tolerance information to be included with the geometric information. A technique which considers tolerances as *attributes* of the model is developed in the work described in this thesis. (Chapter 7 and Chapter 8 describe geometric tolerances and their representation within set theoretic models in more detail).

Before the planning of the inspection of a component can occur, a decision has to be made about where to inspect and about the quantity of probing points to use. The decision depends on the geometric shape of the parts of the component to be checked and the tolerances applied (different tolerance types, in general, require a different number of points). This thesis describes techniques of automatically selecting sets of probing points for a toleranced set theoretic model.

Given the set of surface points, a collision-free path is required for the measuring probe. Solid modelling representations can be used for collision detection and an aim of this project is to demonstrate an algorithm which enables collision detection and path-correction for simple CMM probes. This can then be used to automatically generate collision-free measuring paths.

Once the measurements have been collected, tests are required to determine whether the measured features of the object satisfy the tolerances defined in the specification. A toleranced set theoretic solid model will contain the nominal geometrical information of the object together with the tolerance information, and this thesis aims to show how such a model can be used in the analysis of the results and for finding tolerance errors.

The algorithms described in this thesis were designed for set theoretic solid models using planar half-space primitives and measuring probes with three degrees of freedom which is adequate for a large number of applications. Suggestions for extensions are given where possible.

1.8 Outline of Developed System

The purpose of this section is to describe the inspection system developed within this thesis and to compare the techniques with current practices, highlighting where new work has been done. Figure 1.1 displays the developed inspection system as a flowchart. Each step will now be considered in turn.

1.8.1 Representing the Toleranced Component

Currently machined components are designed and represented as two-dimensional drawings consisting of two or more planar views of the component. The diagrams are drawn with dimensions and tolerances using standards described in BS308. In most cases a Computer Aided Design package is used to help create these diagrams which are then passed onto the inspection team. Figure 1.2 gives an example of a toleranced drawing of one of the components used as a test piece for the work developed in this thesis.

The solid model representation of the machined components is constructed using a computer user interface which enables geometric primitives to be defined and combined. (For the modelling scheme used at Bath University, this interface consisted of a structured Model Description Language based on PASCAL. A compiler, written in FORTRAN, is used to process the source code and generate the set theoretic representation of the component). The work done in this thesis required extending the traditional set theoretic representation scheme so that non-geometric attributes could be added to geometric primitives. This allows the specification of geometric tolerances. Fig 1.3 shows a section of the language used to define the shape in Fig 1.2 and shows how some of the geometric tolerances are defined.

The idea of attributes of geometric entities cannot be as easily implemented within traditional boundary representation models which contain lists of edges, faces and vertices. However most user interfaces to such modelling schemes allow components to be built constructively in the same way as set theoretic models (hiding the underlying boundary structure from the user) and it is possible to label lists of vertices, edges and faces with attributes. This would allow the same scheme developed within this thesis to be used on Boundary Representations. It is worth noting that the set theoretic approach allows the expensive calculation of edges, vertices and faces to be left until needed.

Dimensional tolerances require references to dimensional information and this is not explicitly available within either modelling scheme. A more complicated scheme is required to enable identification of the toleranced dimensions and this has been targeted as a topic for further work.

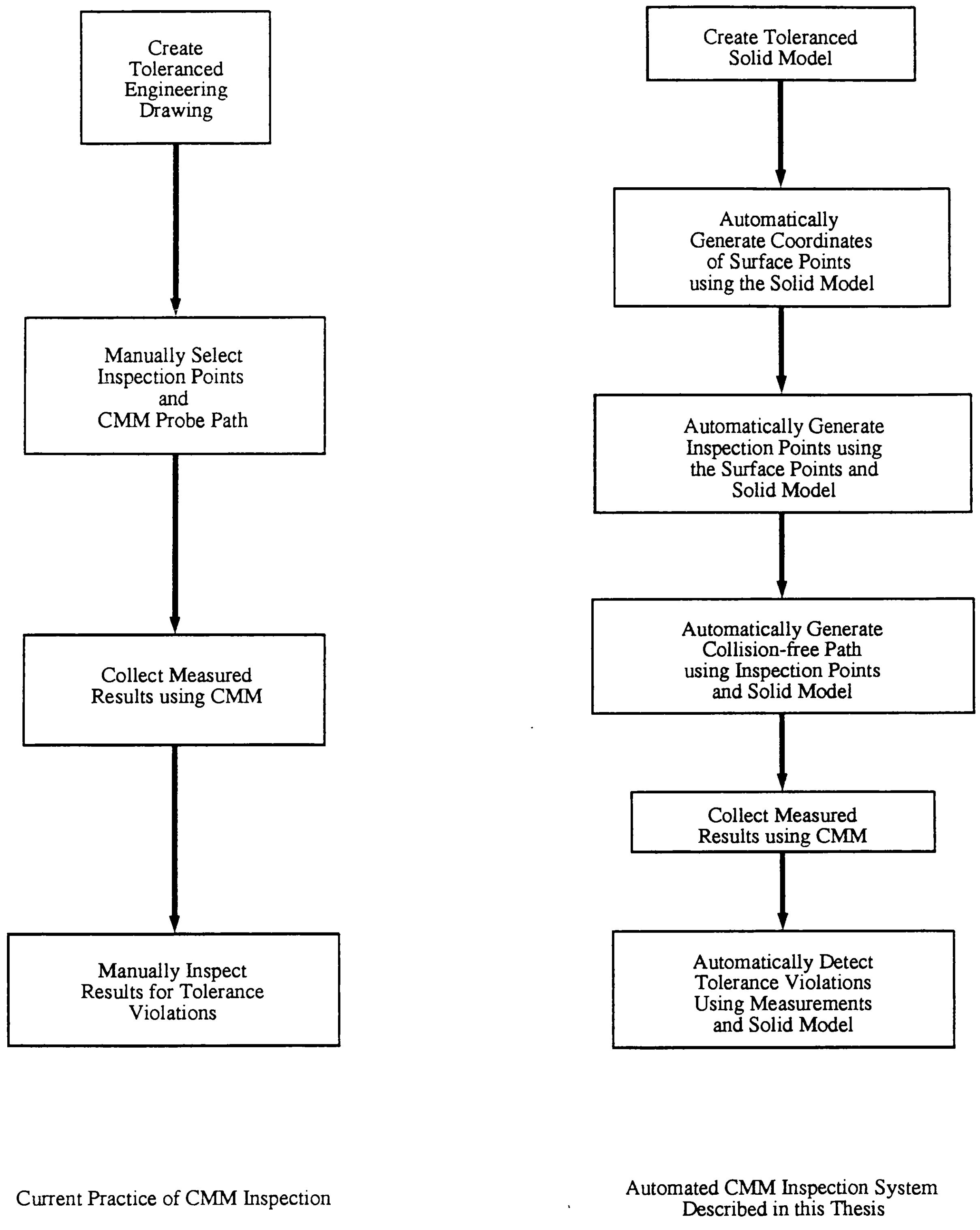

Create Toleranced Engineering Drawing
Manually Select Inspection Points and CMM Probe Path
Collect Measured Results using CMM
Manually Inspect Results for Tolerance Violations
Current Practice of CMM Inspection
Create Toleranced Solid Model
Automatically Generate Coordinates of Surface Points using the Solid Model
Automatically Generate Inspection Points using the Surface Points and Solid Model
Automatically Generate Collision-free Path using Inspection Points and Solid Model
Collect Measured Results using CMM
Automatically Detect Tolerance Violations Using Measurements and Solid Model
Automated CMM Inspection System Described in this Thesis

Figure 1.1 Flow chart of automated inspection system.

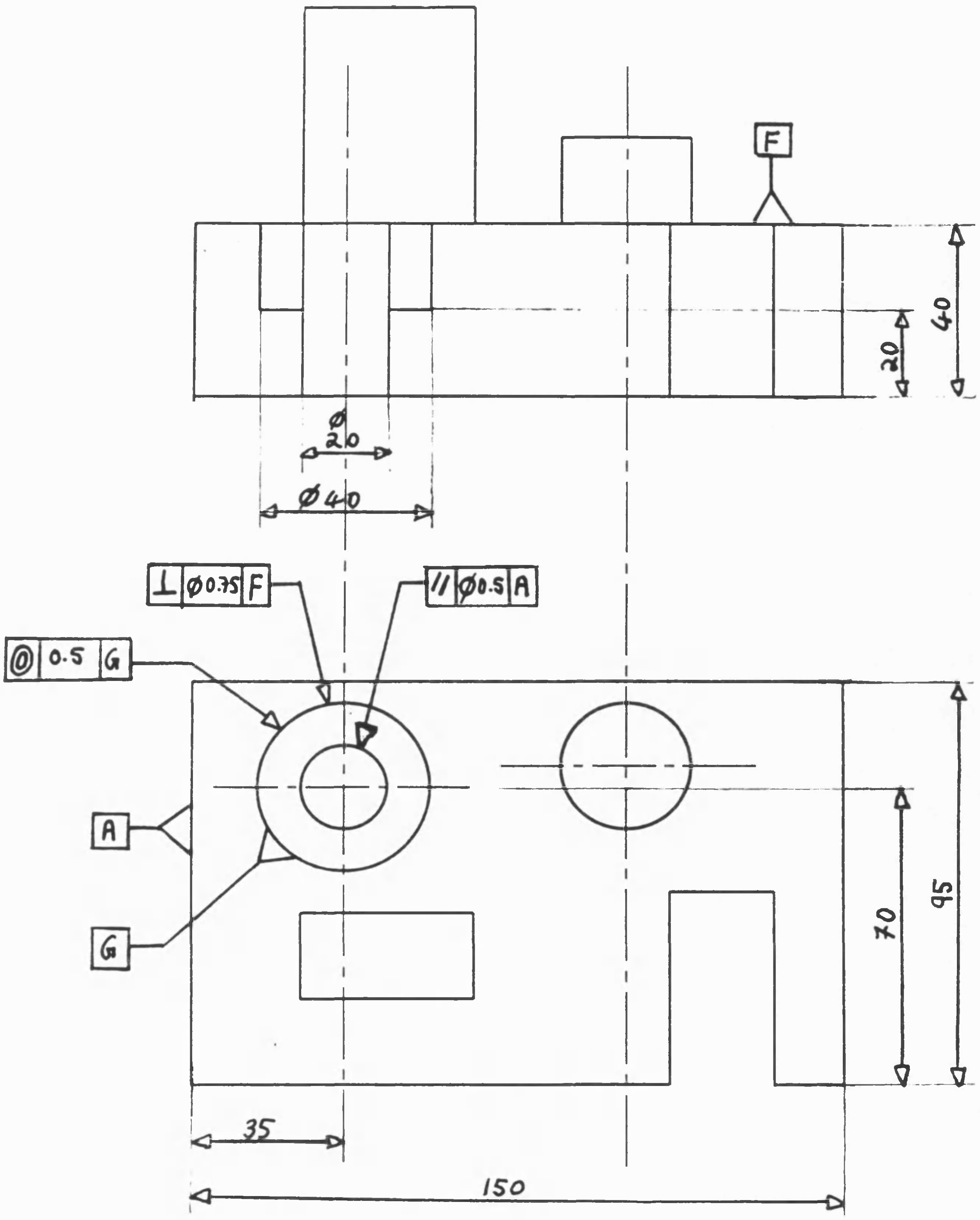


Figure 1.2 Toleranced engineering drawing of component.

```
FUNCTION base_box(lowcorn:Point,highcorn:Point): Set
; Defines the base block and assigns the datums for measuring.
;
Sets {x_space,y_space,z_space,cub_oid}
{
  x_space := space(pt(-1.0,0.0,0.0),lowcorn)
  y_space := space(pt(0.0,-1.0,0.0),lowcorn)
  z_space := space(pt(0.0,0.0,-1.0),lowcorn)

  x_space := attribute (x_space,meas_datum,"X_MEAS_DATUM")
  z_space := attribute (z_space,meas_datum,"Z_MEAS_DATUM")

  x_space := datum (x_space,"A_DTM")
  y_space := datum (y_space,"B_DTM")

  cub_oid := x_space & y_space & z_space

  x_space := space (pt(1.0,0.0,0.0),highcorn)
  y_space := space (pt(0.0,1.0,0.0),highcorn)
  z_space := space (pt(0.0,0.0,1.0),highcorn)

  y_space := attribute (y_space,meas_datum,"Y_MEAS_DATUM")

  x_space := datum (x_space,"C_DTM")
  y_space := datum (y_space,"E_DTM")
  z_space := datum (z_space,"F_DTM")

  cub_oid := cub_oid & x_space & y_space & z_space

  RETURN (cub_oid)
 }

 ;
 ; Build stepped through hole
 ; Centre (x=35,y=70)    ; Outer Radius = 20 ; Depth = 20 ;
 ;                       ; Inner Radius = 10 ; Depth = 40 ; Diff
 ;
   cylrad := 20.0
   radprop := cylrad/dev
   nfacets := cylfacetnum (cylrad,radprop)
   temp_cyl := cylinder( ln (z_dir,pt(35.0,70.0,150.0) ) ,cylrad,nfacets)
   temp_cyl := temp_cyl & space (z_dir,pt(35.0,70.0,45.0));
   temp_cyl := temp_cyl & space (-z_dir,pt (35.0,70.0,20.0))

   temp_cyl := colour (temp_cyl,3)

   temp_cyl := attribute (temp_cyl,gen_feat,
         "CYL_AXIS : D0.0 0.0 1.0 R20.0 P35.0 70.0 150.0");
   temp_cyl := datum (temp_cyl,"G_DTM")

 ;
 ; axis - plane sq
   temp_cyl := tol_set (temp_cyl,SQUARENESS,0.75,pt(0.0,0.0,0.0),"-DF_DTM")

   temp_cyl := attribute (temp_cyl,inspect,"I10.0")
   fyr_demo := fyr_demo - temp_cyl
 ;
   cylrad := 10.0
   radprop := cylrad/dev
   nfacets := cylfacetnum (cylrad,radprop)
   temp_cyl := cylinder(ln (z_dir,pt(35.0,70.0,150.0)),cylrad,nfacets)

   temp_cyl := temp_cyl & space (z_dir,pt(35.0,70.0,25.0))
   temp_cyl := temp_cyl & space (-z_dir,pt (35.0,70.0,-5.0))
   temp_cyl := colour (temp_cyl,4)

 ;
 ; axis - plane par
   temp_cyl := tol_set (temp_cyl,PARALLELISM,0.5,pt(0.0,1.0,0.0),"-DA_DTM")

   temp_cyl := attribute (temp_cyl,inspect,"I5.0")

   temp_cyl := datum (temp_cyl,"H_DTM")
   temp_cyl := attribute (temp_cyl,gen_feat,
         "CYL_AXIS : D0.0 0.0 1.0 R10.0 P35.0 70.0 150.0")
 ;
 ; axis - axis conc
   temp_cyl := tol_set (temp_cyl,CONCENTRICITY,0.5,pt(0.0,0.0,0.0),"-DG_DTM")

   fyr_demo := fyr_demo - temp_cyl
```

Figure 1.3 Set theoretic solid model description of component.

1.8.2 Generating Inspection Points

The decision about where to measure the component is currently done manually using the toleranced 2-Dimensional engineering drawing. Skilled technicians select inspection points during the teaching and/or programming of the CMM, and base their decisions on experience. As with drawings there are Computer packages which make the task easier.

The work developed in this thesis is novel in that it uses toleranced solid model representations of components to generate the *inspection points*, i.e. the 3-dimensional coordinates of points on the components' surface which are to be measured. The algorithms (written in C) applied to solid models in order to generate these points are specific to set theory representations, being reliant on the existance of component half spaces. However it seems probable that similar algorithms can be developed for the *faces* of boundary representations if required.

There are two parts to the generation of points as implemented in this thesis. Firstly a superset of surface points are generated for the geometric model. Then a subset of inspection points are selected from this. The selected points are dependent on the geometric tolerances applied to the model and the physical geometry of the CMM probe used to measure the component at the inpection points; generally the probe will not be able to reach all of the component surface. (The task of determining which points can be reached is a special case of the collision detection problem described in 1.8.3 below).

1.8.3 Generation of Collision-Free Inspection Path

Once a decision has been made on where to measure the component, an *inspection path* is required for the CMM probe. This is currently done manually by skilled technicians who teach and/or program the CMM, possibly with the aid of computer user interfaces.

A new application of solid model is developed in this thesis by using the models for automatically generating collision free inspection paths. The algorithms (written in C) developed use set-theoretic representations to estimate *grown* models which can be used with ray casting techniques to detect collisions. The grown models are estimated by modifying the existing representation and this is much easier with set theoretic modelling than boundary representation were extra care is necessary to ensure that the modified model will still be a valid representation.

Collision avoidance is only one issue of generating an inspection path. It is clearly advantageous to reduce the time taken for the CMM to collect the measurements and this can be done by keeping the distance travelled by the probe to a minimum. The algorithms developed in this thesis partly achieve this by associating a cost with probe motion and choosing a sequence of motion which minimises this cost. (In practice, however, the cost of probe motion is not only dependent on the distance travelled, but also on the type of probe, i.e. whether it is swivel or not and on the nature of the CMM being used; for example, at Bath University, motion in the y-axis was much faster than that in the x-axis). The principle of having a cost function of probe motion can always be applied, although the nature of such a function is highly dependent on the CMM.

1.8.4 Analysis of Measurements

The CMM collects the 3-Dimensional coordinates of points lying on the component surface. These measurements are then used to determine whether the component has been machined correctly and whether any errors which have occurred are within the permitted tolerance constraints. Currently this is done using software to calculate least-square fits to the surface entities and a user interface to enable comparison of nominal and measured geometric entities. Less sophisticated though more common techniques rely on manual comparison of the measured and nominal parameters.

The work done in this thesis uses the set theoretic solid model representation as the nominal shape and compares the measurements against this model. Surface fitting algorithms are used were necessary to estimate the parameters of the toleranced geometric entities. Algorithms have been developed to calculate the *Tolerance Zones* associated with the toleranced entities and tests are made to check that the entities defined by the measured parameters lie within this zone. (Tolerance Zones are specified in Standards BS308 Part 3 although the work in this thesis identifies a failing within these specifications).

1.9 Layout of This Thesis

Chapter 2 describes the current literature and work done on related topics. The current status in automated inspection, solid modelling, collision-free path planning and attributes in solid models are reviewed in this chapter.

To generate inspection points automatically three problems have to be solved. Firstly the generated points must lie on the surface of the component and secondly the points must be reachable by the probe. Thirdly the points must be sufficient in number and in layout for the inspection task. In view of this, the task of generating inspection points has been split into two sections. Chapter 3 describes methods of generating the coordinates of model surface points. An algorithm is described for models which use only planar half-spaces and a separate algorithm is given for curved surface models.

Since probe access is a consideration in selecting inspection points and this requires a collision avoidance algorithm, Chapter 4 describes a technique of growing set theoretic solid models which is useful for collision detection. This chapter also illustrates how model growing is used for collision detection for simple probes.

Chapter 5 describes techniques of selecting a set of inspection points from a set of surface points. The points are tested for access by the probe and a selection of inspection points is made based on the nature of the feature and the tolerance of the surface containing the point.

Chapter 6 describes methods of using the grown and ungrown modes in generating a probe path which passes through the inspection points. The problems of collision detection and avoidance are considered here as well as the problem of reducing inspection time.

Chapter 7 describes a method of assigning non-geometrical information about geometric shapes, or *entities*, within the component by assigning attributes within set theoretic models. Geometrical tolerances are given as an example of such attributes. The algorithm was implemented using a model definition language written at Bath University and this is outlined in this chapter.

Chapter 8 describes methods of using the solid models and measured results to determine if the geometrical tolerances have been violated. Feature extraction, tolerance zones and the methods of comparisons between the measured points and the model are described here.

Finally, Chapter 9 offers conclusions and suggestions for further work on all aspects of the project.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

This chapter describes work already done in fields related to the work described in this thesis. There are only a limited amount of publications dealing directly with the applications of solid modelling to automatic Coordinate Measuring Machine (CMM) component inspection. However, there are many papers which are relevant to the various separate stages of the project.

2.2 Automating Component Inspection

2.2.1 Motives for Automation

In justifying the automation of inspection by CMMs, three major reasons have been suggested over the last decade. In 1980, R. Gilheany [1] highlighted the extra burden being imposed on inspection departments by new numerically controlled machining techniques. These meant machining was being completed much quicker than inspection. The same problem was also spotted by S. P. Black [2] in the same year. It was clear then that some means of decreasing the time required for inspection was required, the benefits being a higher inspection rate and hence potentially better quality control.

With the development of automated flexible manufacturing systems (FMS) a second reason arose for automating inspection. In a recent paper published in

1988, J. A. Bosch [3] describes how the trend over the last few years has been towards using coordinate measuring machines operated by a host computer. He states that this has been a direct result of the demand for flexible inspection. Many companies, including Rolls Royce, Bendix Ltd and IBM, have on-line coordinate measuring machines controlled automatically by a computer as part of a flexible manufacturing cell, the computers being used both to drive the CMM around pre-programmed inspection paths and to collect and analyse the results of measurement.

J. Raja and U. P. Sheth [4] offer a more strategic motive for automating inspection. They state that inspection can become an effective tool for process control by providing appropriate feedback into the process. They go on to say that this requires a systematic method of relating manufacturing errors to machine tool and process parameters which will lead to a greater understanding of the process. The idea is still in its infancy, with currently only limited in-process gauging being attempted to provide local feedback, but will become more realisable as inspection becomes automated.

In conclusion, motives for automating inspection are the need for speed, flexibility, rapid feedback of results for process control and increasing emphasis on high quality manufacturing.

2.2.2 Current Status of Automatic Inspection

Computer controlled inspection is already used heavily in industry. However, little work appears to have been done on automating the planning of inspection.

J. Raja and U. P. Sheth [4] describe a system which helps develop inspection plans using an inspection database in addition to a CAD/CAM drawing database. The CAD/CAM database contains information about components as two-dimensional drawings and the inspection database holds inspection sequences for various features. A component inspection sequence is built up from a file containing a representation of the component features. The system they describe plans inspection sequences for the checking of several tolerance types, such as out-of-roundness, roughness and dimension.

A similar approach to inspection planning is used by H. D. Park and O. R. Mitchell [5]. In their paper, they proposed the development of a rule-based system which would automatically generate an inspection procedure given design features and feature combinations. The inspection plans generated would be for a computer vision inspection system.

The biggest drawback to such planning is the need to update the database and rule-base as features are changed or new features introduced. They state that in many cases, this can be avoided by slightly modifying the derived inspection plan to accommodate the small changes for features not already represented in the database.

There is much literature covering the automation of the planning of computer vision inspection systems. This type of component inspection, involving cameras and image analysis algorithms, is a growing field. In their paper mentioned above, H. D. Park and O. R. Mitchell describe an inspection planning system which uses boundary representation solid models to relate the camera image features to design features.

Despite the attraction of computer vision inspection, there are still some problems which hinder accuracy of measurement. M. P. Groover and E. W. Zimmers [6] mention three problems. Firstly, the resolution of the image has to be high enough to enable measurements to be made at an acceptable level of accuracy. Secondly, recognising light and dark areas of a component imposes limitations on lighting requirements. Finally, there are limitations imposed by the feature recognition ability.

N. Atkins and S. Derby in their paper, describe a system which automatically generates an inspection program for CMMs using solely data input from a CAD/CAM database [7]. The system is designed as part of a low-cost high-speed computer vision system to make low-accuracy tolerance inspections which are complimented with CMM measurements of the more critical dimensions. As described in their paper, the technique uses a mixture of graphically-aided manual input and predefined macros to develop inspection paths. The macros take geometrical parameters as arguments and return an appropriate path for the feature. Although the system does contain a boundary representation of the component and its features, this is used only for graphical representations and not in the generation of the inspection plan.

This approach has the advantage of enabling a graphical user-interface to the task of inspection planning. However, the path-planning is dependent on the operator, who may still be required to modify the probe path manually and who also has to make corrections for collisions or time wasting moves which arise in the resulting path. In fact, the collision detection itself is done manually by viewing the path via the user interface.

No literature has been found which describes the use of solid modelling representations to automatically generate inspection plans for CMM inspection.

2.3 Coordinate Measuring Machines

2.3.1 Benefits of Inspection by Coordinate Measuring Machines

Manual CMM inspection is very rarely used today. Gilheany and Treywin list three main advantages of computer controlled CMM inspection [1] . Firstly, quicker inspection has led to a greater throughput and cost savings. Secondly, automatic component alignment has meant that there is no requirement for component jig and fixtures, which need to be designed, calibrated and stored. This advantage was also recognised by B. Liddle [8]. Finally, rapid collection and analysis of inspection results means information can be quickly available for processing.

M. K. Groover compiled a list of benefits in his report [9] in which speed is mentioned. This list also includes flexibility and Groover highlights the fact that CMMs can inspect a variety of different part configurations with minimal changeover time. He goes on to state that not only are CMMs inherently more accurate and precise than the traditional manual techniques used, but they are more consistent and, when connected to computers, are less prone to human errors.

In summary, the main advantages of computer controlled CMM inspection are speed, flexibility, reduced cost and accuracy. The machines also simplify the task of integration into manufacturing systems, as mentioned by various authors, such as J.A. Bosch [3] and I. Bowman [10].

2.3.2 Current Coordinate Measuring Machine Technology

The main development of CMM technology has been the increase in computer sophistication. A recent survey carried out by the journal Quality Today [11] shows that the standard of software supplied with CMM systems is considerable, including graphic display of inspection results, three dimensional contour analysis of surfaces, and FMS link facilities. The development of user-friendly software packages for CMM inspection is also a current trend, with some systems using graphics to help in developing inspection paths and in displaying the results of measurement.

Work is being done on integration of CMM into flexible manufacturing systems. A. Sostar [12] describes techniques of integration, and concludes that problems which arise during production and measurement processes can be remedied as CMM software technology improves. Integration is also encouraged by hardware improvements, and a recent development has been automation of loading and unloading of components onto CMMs.

In the production of CMMs, higher accuracy, speed and low cost seems to be the major trend. Touch sensitive probes with greater sensitivity and higher sophistication are being developed, and the machines themselves are being designed to withstand shop-floor environments.

In summary, the current trends in CMM technology are improved computer sophistication and more robust hardware.

2.4 Solid Modelling

2.4.1 Development of Set Theoretic Solid Modellers

There is much literature available on solid modelling techniques, which have had wide applications in computer graphics before being used in CAD/CAM systems. It is generally agreed that most solid modellers fall into two categories; boundary representation and set theoretic (also known as constructive solid geometry (CSG)). A. Requicha describes and compares both types in his 1983 reviews on the history and research of solid modelling [13], [14]. The reviews also illustrated that set theoretic modelling was not as widely used as boundary representation, which appeared to be the nature of many available commercial solid modellers.

In 1983, A. Requicha [15] described the necessary characteristics of a representation of rigid solids. These included formal properties, such as validity, unambiguity and uniqueness of representations, and informal properties such as conciseness, ease of creation and efficacy in the context of application. C. Brown [16] offers similar classification in his paper describing the mathematical and representational aspects of solid modelling. Set theoretic modellers have the advantage of satisfying the validity, unambiguity, ease of creation and conciseness properties, and so would seem to be a natural choice for solid representation.

However, the drawbacks with set theoretic representation, such as non-uniqueness and not having geometrical information readily available, was seen as a disadvantage for most applications. K. Howard [17] offers a list of advantages of boundary representations over set theoretic which have played a part in their acceptance. His list include the fact that boundary representation is in keeping with the historical tendency of designers to think in terms of object edges and intersections. (A more likely reason is the fact that they were well understood from research in computer graphics, with set theoretic modelling being a relatively new idea.)

The problems with boundary representation are well documented. Such modellers need a large and complicated database to represent objects and also a set of extra rules to maintain consistency. The former is the storage for edges, vertices and faces and the latter are combination rules, such as Euler's formulae [18] . Thus, research has been encouraged into overcoming the problems of set theoretic modelling.

In a recent paper, J.R. Woodwark (1989) [19] summarises the progress made over the last ten years. Object space sub-division algorithms [20] & [21] have been developed to considerably reduce time taken on processing models. These have the effect of spatially localising information and increasing the efficiency of model rendering algorithms.

Localisation of information within set theoretic models is the major concern of the work done on the theory of constituents [22] and also of the work done on active zones [23]. These papers also address the non-uniqueness of set theoretic expressions which leads to problems such as creation of over-complicated representations of objects, and the hindering of any subsequent feature recognition algorithms.

Many objects contain blends between surfaces to form features such as fillets. One method of representing such blend surfaces is by using algebraic surfaces [24] . These representations are much simpler than the traditional method of parametric patches. Much work has concentrated on way of determining the algebraic equation of the blend from the equations of the surfaces between which the blend occurs. Woodwark describes a method of blending based on Liming's formula [25] , A. Middleditch and K. Sears [26] offer an alternative formulation, and recent work has been done by J. Warren [27]. One advantage of representing blends as algebraic surfaces is that it allows blends of blends to be easily generated, and clearly increases the range of shapes representable by set theoretical modellers. Further, current developments in computer algebra techniques encourage the use of implicit algebraic representation of shapes [28].

2.4.2 Applications of Set Theoretic Solid Modellers

There are many engineering applications where visualising three dimensional components is advantageous, and the earliest uses of solid modelling was in the generation of pictures. J. Woodwark and A. Bowyer describe algorithms for generating pictures from set theoretic models using ray casting techniques which have better than linear time complexity behaviour [29]. The algorithms use spatial division and model pruning methods, and the paper shows several examples of forms of output including wire-frame and shaded pictures, and the generation of surface texture using fractals.

Extracting other information from solid models, such as volume and mass properties, are also mentioned in the above paper. A. Wallis goes a step further and describes procedures for obtaining other data from solid models [30] . In his paper, he describes an interrogator which extracts dimensional data from solid models, using an interactive interface which provides similar software tools to engineer's hand measuring tools.

J. Woodwark suggests that solid modelling has applications to cutting tool-path verification [31] and in his thesis, A. Wallis describes methods of doing this [32]. He describes algorithms for generating volumes swept by cutting tools as they move along tool-paths. These may be differenced from a model of the blank uncut block, and the resulting shape may then be interrogated to assess the accuracy of the path.

Sweeping solid models has other applications, such as mechanism design and robot path planning, where analysis of interference between moving objects is required. R. R. Martin describes determination of sweep volumes using differential geometry theory [33] . Interference checking is the process of deciding whether there is a volume of solid which is common to two or more solid models. This can be done by intersecting the two models and deciding if the resulting model contains solid or not, i.e. is null. For boundary representation, the intersection calculation is difficult, but the subsequent null object detection reduces to the test of whether or not there is a boundary in the resulting shape. For set theoretic representation, intersection is trivial but null object detection becomes a problem, with many existing algorithms resorting to calculation of boundaries [34].

So far, boundary models have been favoured in the field of feature recognition. The recognition of geometric features within models has obvious applications in process planning and other decision making systems. J. Woodwark describes the problems of both forms of modellers in his speculations on the recognition problem [35] , but argues that the robustness and tendency of algorithms for set theoretic modellers to be easily theoretically justified makes them the better choice. However, as highlighted by G. Jared [36] , the problem of feature recognition seems to be strongly dependent on application and thus probably cannot be solved using only the geometric information present within the model. This seems to suggest a need for assigning more information to the solid model, such as non-geometric attributes, which provide the necessary data for unique recognition. Indeed, such an approach is used by D. Perng, Z. Chen and R. Li in extracting machining feature information from set theoretic models [37] . They build set theory trees with primitive nodes containing extra information about the primitives, which is used for recognition.

There are many other applications of solid modellers being investigated. P. Sabella and I. Carlbom describe applications in modelling oil reservoirs [38]. V. Alagar, T. Bui and K. Periyasamy are amongst a group of researchers investigating the automatic generation of finite element meshes using CSG models [39].

There has been little work done on applying solid models to the task of inspection planning. In their planning of visual inspection, D. R. Mitchell and H. D. Park use a boundary representation solid model to link database feature information to the features recognised from a camera image, and thus enable comparisons between the ideal and measured component. However, the inspection plan and results analysis is done independently of the model. Considerable work has been done on introducing non-geometric information into solid modellers, such as tolerance information [see Section 2.8], but little has been done on the subsequent application of such models to inspection planning.

In summary, solid modelling is still finding new applications in a large range of manufacturing areas, but has yet to be fully applied to the task of inspection planning.

2.5 Generating Surface Points

2.5.1 Generating a Regular Grid of Points

Little work has been done on using solid model representations to determine a set of points lying on the surface of objects. The closest work has been in generating finite element meshes.

S. Tan and M. Yuen describe an algorithm which uses spatial division techniques to generate meshes [40] . In their work they describe a spatial division algorithm together with procedures for clipping the resulting boxes to surface and maintaining continuity of the mesh. In theory, the mesh nodes obtained from the division sub-spaces can be used as surface points.

V. Alagar, T. Bui and K. Periyasamy have also considered the problem of generating meshes from set theoretic solid models [39]. They describe a method which builds the mesh constructively by combining meshes already existing for primitives. The paper describes how joining of meshes is done, but does not describe how the original meshes are calculated.

The main drawback with this approach being applied to point generation is that the meshes, although variant in selected regions, are essentially regular and thus the resulting set of points may miss cyclic deformations, i.e. deviations which occur at regular places throughout the measured surface.

This suggests that these finite element mesh generation techniques are not suitable for the generation of inspection surface points.

2.5.2 Interval Arithmetic

Chapter 3 of this thesis describes an algorithm which is used to generate surface points on models containing curved surface half-spaces. The algorithm needs to calculate estimates of the range of normal direction of the curved surfaces over cuboid boxes of differing position and size, and to do this, *interval arithmetic* is used.

The subject of interval arithmetic has been well documented. R. Moore has written several definitive books offering rigorous analysis of the theory of interval arithmetic [41] . The arithmetic allows conservative estimates on the range of functions whose argument(s) vary over an interval(s). The technique is to define rules of addition and multiplications of intervals, and to evaluate the function using the intervals over which the arguments vary as arguments themselves.

There have been many papers published which describe methods of making the estimates of interval arithmetic less conservative, specifically by considering the various ways of re-writing polynomial and rational functions. S. Pai considers the benefits of various standard forms, such as centred form and mean value form [42] . E. Hansen describes a method of obtaining tighter bounds on function estimates by introducing a new generalized interval arithmetic, which offers a different interpretation of intervals [43].

For surface point generation, estimates are required for the range of normal direction of polynomial surfaces within cuboids. The problem of conservative estimates is only significant for third or higher order polynomials, and since a majority of component surfaces are quadric, resulting in linear normal equations, little or no improvement is obtained by using the various suggested techniques for obtaining better estimates.

2.6 Reconstruction of Set Theoretic Model Expressions

A method of detecting collisions between moving objects which relies on the *growing* set theoretic solid models is described in Chapters 5 and 6. Chapter 4 of this thesis describes how reconstruction of set theoretic expression enables set theoretic models to be grown. Model reconstruction as a technique is not new and this section describes research done in this area.

Set theoretic model reconstruction involves transforming the set theoretic expression to another form. The new expression still represents the same object but is more relevant to the particular application.

J. Goldfeather, S. Molnar, G. Turk and H. Fuchs [44] use model reconstruction techniques to efficiently generate pictures from solid models. They argue that normalizing a set theoretic expression to a union-of-intersections expression enables faster rendering in depth-buffer algorithms. The algorithms are developed for bounded primitives.

The normalization algorithm involves using equivalence mapping of expressions, and the authors mention that combinatorial explosions in the size of the resulting expression is a potential problem. To overcome this, they suggest techniques of geometric pruning based on bounding boxes. Although these prove to be adequate for bounded primitives, where bounding boxes are well defined, no simplification methods are given for handling unbounded primitives such as planar, or higher degree half spaces. If such half spaces are used in a model, normalization could lead to a large increase in the size of the set theoretic expression, and hence greatly hinder subsequent processing.

Y. Lee and K. Lee use set theoretic tree reconstruction to make feature unification and recognition easier [45] & [46] . They claim that the problems of non-uniqueness and the lack of localisation within the tree can be overcome by moving tree primitive nodes such that the expression still represents the same object, but feature primitives are "closer" together within the tree. The moving of nodes is suggested as an alternative to duplication within the tree. Again, the rewriting algorithm depends on equivalence mapping.

The modeller for which the algorithms are developed is based on bounded primitives. Each primitive is stored as a set of attributes describing the geometric nature of the primitive, and this information is used in deciding where to move nodes. The algorithm, therefore, cannot cope with more general models constructed out of simpler primitives, such as planar half spaces. These simpler primitive modellers allow more flexibility, but increase the non-uniqueness of representations by introducing numerous new ways of describing a feature.

In summary, set theoretic model reconstruction is being used to attempt to solve several problems and seems to be more successful on modellers which use bounded primitives. These modellers provide a little more information than those which use unbounded primitives but add complexity by introducing redundant surface segments.

2.7 Selection of Inspection Points

The selection of sets of inspection points for checking features is currently performed manually. As described in Chapter 1, this requires a certain amount of skill. Little work has been done on automating this task.

The British Standard BS7172, however, offers suggestions for choosing inspection points for common geometric shapes such as planes, cylinders, spheres and cones [47]. As the standard states, no absolute guidance is possible because the chosen points must take into account the machining process and intended function of the component.

For planes, the standard suggests aiming for uniform coverage to ensure a genuine representation, but also suggests avoiding regular spacing which would miss periodic surface deformations.

For spheres, the standard considers distributing N points on the surface of a sector of a sphere, radius r, enclosed between two parallel planes, separated by distance h. This is done by calculating n_c parallel planes and distributing N/n_c points on each of the n_c circles which are obtained by intersecting the planes with the sphere. The value of n_c is calculated as the nearest integer to:

$$\left\{ \frac{Nh}{(2\pi r)} \right\}^{\frac{1}{2}}$$

The standard goes on to suggest ways of avoiding regularity.

For cylinders, a similar approach is suggested, with alternate odd and even number of points on each circle again to reduce regularity.

Finally, for cones a cone section is considered between two planes, a distance h apart and with their normal directions aligned with the axis of the cone. The circles of intersection of these planes with the cone have radii r_1 , r_2 where $r_2 > r_1$. The value of n_c is now taken as the nearest integer to:

$$\left\{ \frac{N\{ h^2 + (r_2 - r_1)^2 \}^{\frac{1}{2}}}{\{ \pi(r_1 + r_2) \}} \right\}^{\frac{1}{2}}$$

An extra parameter, s is calculated and this is the reduction in number of points to distribute on each circle as the cone apex is approached. This parameter is estimated from:

$$s = \frac{2\pi(r_2 - r_1)}{\{ h^2 + (r_2 - r_1)^2 \}^{\frac{1}{2}}}$$

The guidelines are sufficient for manual inspection, where the operator can adapt the suggestions to the particular component being inspected. However, they do not help in automating the technique since such reasoning is difficult to incorporate within algorithms which use solid models. This is because the variable h in the above equations is not readily available within the model. Another problem is that, although the circles can be calculated for the quadrics as specified, not all circles may actually lie on the component surface since only part of the quadric half space may contribute to the real surface of a model. Hence success depends on how closely the features within the component resemble the complete primitives for which the points have been chosen. Further, points on some of the circles chosen may be impossible to reach with the inspection probe.

The standard, however, does offer useful information on choosing the quantity of points which is dependent on the tolerance being checked. Table 2.1 summarises the suggested quantities. A basic rule is that tolerances of form, such as roundness and flatness, require more points than tolerances of location and attitude, such as position and parallelism.

Feature	Tolerance Type	Minimum Points	Recommended
Plane	All Types	3	9
Sphere	All Types	4	9
Cylinder	Attitude Location	5	12
	Form		15
Cone	Attitude Location	6	12
	Form		15

Table 2.1: Suggested and minimum quantity of inspection points for various features and tolerance types.

2.8 Path Planning

2.8.1 Efficient Paths

The problem of driving the probe around a set of three-dimensional coordinates in such a way as to minimise the total distance travelled is related to the travelling-salesman problem, where a salesman has to visit a number of cities and spend minimum time travelling. The problems are not identical, since the CMM probe is not constrained to using particular pre-determined paths whereas the travelling salesman must use roads. (There are thus less constraints on the CMM probe then there are on the salesman). The travelling-salesman problem has been the topic of much research, finding applications in a wide range of areas.

The major trend in the work has been to develop algorithms which solve the problem quickly. The problem has been proven to be *NP-hard* and thus can be solved exactly in computation time which is of exponential order. For cases which require an exact solution, algorithms have been developed which depend on branch-and-bound methods used in operational research. The idea is to successively divide the possible solutions into smaller sets, which have a lower bound on the cost of the solution, until a set containing one solution is found. J.Little *et al* describe the technique, which uses a cost matrix approach, in detail [48] . The computation time for various cases is given in the same paper and seems to suggest that 40 cities is an upper bound on the size of the problem it can handle within reasonable time. (The authors go on to state that adding ten cities results in a about a ten-times increase in computation time).

An exact solution is not always required, and in many cases one which is in some sense close to optimum will suffice. In view of this, several heuristic algorithms have been developed to solve the problem. These involve starting with a known initial solution consisting of an ordering of cities, and making changes in this ordering which reduce the cost. Eventually a minimum, which may be global or local, is found when no further changes can be done to reduce the cost. S. Lin and B. Kernighan describe an algorithm which attempts to makes as many changes to the ordering as it can at each iteration [49]. E. Aarts *et al* suggest a simulated annealing approach where the number of changes made at each stage reduces in subsequent iterations [50]

The heuristic algorithms have limited use for problems with more than 100 cities, although success is very dependent on the initial solution chosen and the nature of the problem. From a probe path planning point of view, where the cities become inspection points on the component surface, the expected number of points can vary from 30 to 300, and any restriction on the number of points should be avoided. Further, the costs of moving the probe from one point to another is a non-trivial calculation.

Travelling-salesman algorithms require the calculation of all costs before searching for a solution and this can be a time-consuming exercise in itself. In view of this, and the unpredictability of success with heuristic algorithms, the probe path problem has not been considered as a case of the travelling-salesman problem in the work described in this thesis.

The nearest-neighbour problem is defined as the problem of finding points in a set which are closest to some fixed point. Several algorithms exist for doing this and P.Vaidya describes an algorithm for the all-nearest-neighbours of a point in a set of size n points in time which is of the order $n \log n$ [51]. The algorithm uses recursive division methods to obtain a set of spatially related boxes, each box containing a single point. Voronoi diagram methods provide another means of calculating nearest-neigbours since these points are exactly the Voronoi neighbours.

A path can be constructed by selecting the nearest neighbour to a point at each stage. Although non-optimal, the nature of such a path can be controlled more easily than those paths generated by travelling-salesman methods. In some cases, for example, a group of points may be required to be inspected together, and this can be done trivially using attributes and the nearest-neighbour method of path planning.

2.8.2 Collision Detection

As the CMM probe moves around the component taking measurements, collisions between the probe and the component should be avoided. Thus some means of detecting collisions during path planning is required. The path consists of an ordered list of three-dimensional coordinates and the collision detection algorithm needs to be able to decide if the probe can move along straight lines between adjacent points without colliding with the object.

If the volume swept by the probe in moving between the two points is modelled as a set-theoretic expression, then the problem becomes one of assessing whether or not the set-theoretic expression derived from intersecting the component representation with this swept volume representation is null. The construction of swept volumes for the probe is easy since the probe only consists of cylindrical and spherical elements. However, null-object detection is a non-trivial problem and often involves calculation of the object boundary. R. Tilove describes methods of spatial localization and primitive redundancy detection which lead to more efficient null-object detection algorithms by only evaluating boundaries when necessary [34].

V. Alagar, T. Bui and K. Periyasamy describe applying the principles of solid modelling and sweep volumes to robotics [52]. They develop a reasoning system based on a solid modelling representation and describe methods of detecting collision, using intersection calculations and null object detection, and methods of controlling assembly operations.

In his paper, S. Cameron describes a method for detecting whether an object is null or not. The approach starts with bounded primitives and propagates the bound values throughout the set-theoretic expression to yield bounds on the component and its various component sub-models. These bounds are known as *S-bounds*, and having calculated the S-bounds for the model, Tilove's methods are applied again to detect null-volume. The S-bounds help reduce computation and, in some cases, are enough to detect null-objects [53].

S. Cameron also describes a method of collision detection which uses a notion of extrusion into space-time to model motion, and describes methods of interference detection based on models of these extrusions [54] . The technique is superior to sweeping since the relative state of the moving objects can be extracted from the extrusions at any given time, and this information is lost in swept models. However, if only one object is in motion this offers no advantages from a collision detection point of view.

C. Mirolo and E. Pagello describe a technique for collision detection using a boundary representation solid modeller [55]. The approach is restricted to using polyhedral primitives and involves iterative search algorithms to determine whether an object will collide with another when moved along a given direction. The technique is computationally complex and involves consideration of various cases in order to account for all possibilities. However, the paper claims a run-time of order $\log^3 n$ for n primitives and the approach does have the advantage of yielding information about the intersection, such as which face of a polyhedron is hit, and this is useful for algorithms which avoid collisions by automatically moving around obstructions.

K. Kondo and K. Kimura describe an alternative approach to collision detection which involves calculating the available free-space within a manipulator work space [56]. The method uses a grid-based wave-front propagation technique and models the obstacles and manipulator approximately using a boundary representation solid modeller. The algorithm successfully handles a large number of manipulator shapes and could be used for CMM probe path-planning. However, the computational cost for such a general approach suggests developing methods which can take advantage of the simple geometry of most CMM probes.

T. Lozano-Perez and M. Wesley describe an algorithm for planning collision-free paths among convex polyhedral obstacles which uses the concept of growing the obstacles to create forbidden regions, and in using vertices of these regions to derive the path [57]. The obstacles are represented as vertices and the forbidden regions are approximated by translating these vertices appropriately. The paper gives a useful insight into the problems of growing when considering the motion of general polyhedral shapes, includin rotations as well as translations. They extend their ideas to introduce a *configuration space* approach in a later work [58]. In this paper, configuration space obstacles are defined as the geometric objects which represent all the positions of object A that cause collisions with the object B. The configuration space obstacles are constructed by selecting a vertex of object A and calculating a *difference* set between the set representing object B and the selected vertex of A. The paper proves that if the selected vertex of A lies within this new set, then the intersection between the two objects is non-null; i.e. collision does occur, and this enables the problem of collision detection to be reduced to one of determining whether a point lies within a set. The paper offers rigorous analysis for the two-dimensional case and demonstrates how the ideas can be extended to three-dimensions, where the configuration space obstacles are approximated.

Chapter 4 describes a method of *growing* set theoretic models of components to obtain new models which can be used for the detection of collisions between a CMM probe and a component. The probe is modelled as two cylinders and a sphere, and a grown model is constructed for each of these probe parts. The grown model is an approximation to the configuration space obstacle which results from selecting the centroid of the (symmetrical) probe parts, rather than a vertex, and calculating the difference set as explained above. Collisions can thus be detected by testing these centroid points against the corresponding grown model.

The Lozano-Perez algorithms described above uses graph search techniques to determine an appropriate path for the obstacle A. A similar graph-based approach is used by L. Fu and D. Liu, who describe a method of determining the minimum-distance collision-free path using a visibility graph constructed from the obstacle vertices [59]. Such methods work well for two-dimensional path planning, but in three-dimensions, the shortest-distance path is no longer guaranteed to pass through vertices. This problem is overcome by calculating approximate solutions using "phantom" vertices introduced within edges. The closeness of such approximation depends on the locations of these extra vertices.

2.9 Tolerance Representation and Definition

2.9.1 Assigning Tolerances to Solid Models

In their paper, J. Shah and D. Miller state that a geometric model defined in terms of low level geometric and topological entities or primitives contains insufficient information for the geometric reasoning required by most CIM programs. [60]. They offer a list of applications which require extra information and this list includes group technology coding, variant and generative process planning and NC toolpath generation. Their paper goes on to describe a component part definition system consisting of a dual solid modeller (i.e using both boundary and set theoretic representations) to define the nominal geometry; a feature model to capture the semantics of the geometry; and variational attributes to specify tolerances and surface finish. The scheme still relies on attaching non-geometric information directly to the solid model in order to allow communication between these three parts of the system.

The problem of assigning tolerances as attributes to set theoretic solid modellers was considered by Requicha [61]. He describes a method which uses a variational graph which is related to the faces of primitives in the solid model and which contains variational non-geometric information such as tolerances. The graph allows the relationship between the various features to be established and uses an indexing scheme for naming primitive faces based on the position of the primitive within the set theoretic expression. Algorithms are described for building the graph incrementally as attributes are assigned to the appropriate features.

The method allows limited editing of the set theoretic model after the application of attributes. However, there is a large memory overhead in using the graph structure, primarily due to the indexing scheme required to localise primitive faces within the set theoretic expression.

The work highlights the need for being able to define simple features, such as planes, cylinders and axes, and distinguishes between entities, such as axes, and the measured entities. Axes in this scheme are defined in terms of symmetric features, such as prisms, cylinders or cones, with extra information to remove ambiguities.

Tolerances are checked in terms of tolerance zones which are built by shrinking appropriate model features and subtracting them from their expanded versions.

Although this approach is directly applicable to form tolerances, such as roundness, no indication is given as to how such a zone is constructed for non-measured entities such as axes.

U. Roy and C. Liu develop a method of tolerance representation which uses a hybrid structure of set theoretic and boundary representation together with a representational tree of attributes [62]. They argue that the set theoretic description is useful for establishing the spatial constraints between features and that the boundary representation allows easy access to the "low-level" feature information. Various graph structures are used to maintain relationship between the features and attributes. The method makes it mandatory to include boundary representations of components and primitives and thus is expensive computationally and in memory.

In representing non-measurable entities, such as axes, it is assumed that other entities exist which define them, such as two intersecting planes. This leads to an unnecessary, and unwarranted increase in the size of the model since the non-measurable entities are implicitly defined by existing measurable entities.

P. Ranyak and R. Fridshal also describe a method for assigning attributes to solid models [63]. The paper describes the construction of a "Dimension and Tolerance Model" in conjunction with a boundary representation of the component. However, the paper does not make it clear how this additional model is used in conjunction with the solid model.

N. Bernstein and K. Preiss also discuss representation of dimensional and tolerance information within boundary representation solid modellers [64]. Little is said about the details of attribute assignment, with the authors mostly describing in detail the nature of the tolerance information. The method seems to adopt the structure of boundary representation but redefines the constituent features, such as faces, edges and vertices, to include extra information about constraints. The paper does offer rigorous analysis of tolerance representation, particularly size tolerances, and defines a method of calculating constraints imposed on the boundary features by tolerances.

The representation schemes described above have been developed with the aim of representing dimensional and size tolerance information in solid models. By restricting attention to geometric tolerances, the problem is reduced considerably since the toleranced features are realisable geometric entities. Chapter 7 describes a method of assigning such tolerances to set theoretic solid models.

2.9.2 Definition of Geometric Tolerances

The British Standard BS308 Part 3 describes the definition and interpretation of geometric tolerances currently in use [65]. The most commonly used tolerances are roundness, squareness, angularity and position. A scheme developed for the representation and verification of geometric tolerances must be able to consider these.

A. Fleming offers a useful analysis of determining the total constraint applied to a feature within a component by the application of geometric tolerances to various parts of the component [66]. The paper describes relationships between various tolerance zones, features and datums, and offers a method of determining the constraint applied to a feature using networks of zones and datums. This allows the validity and meaning of such tolerances to be assessed when they are applied, but the paper does not describe how such information helps in checking that the tolerances hold after measurement.

F. Etesami describes a method of tolerance verification using boundary representation models [67] . The tolerance information is used to construct the boundary representation of a component model built from *constructors*. These constructors represent the allowed variation of the nominal feature, and the paper describes how they are built for entities such as axes, for which he assumes a cylindrical form. Tolerance checking involves extracting the appropriate feature parameters from measured data and testing them against the constructors. It is not clear what advantages, if any, this method has over direct results analysis using the solid model.

The maximum-material-condition tolerance is the subject of a rigorous analysis carried out by R. Jayaraman and V. Srinivasan [68] . They formalise the theory of such geometric tolerances in terms of assembly and material bulk requirements and conclude that current industrial practices are inadequate for their alternative, more powerful, tolerance specification. Maximum-material constraints are tolerances used to control assembly and as such require information about the various independent components. The work described in Chapter 7 and Chapter 8 of this thesis deals with single components and geometric tolerances whose application are restricted to these components, and thus such an alternative specification is not needed.

2.10 Extraction of Feature Information

Extracting geometric parameter information, such as axis directions or radii, directly from solid models has received little or no attention, with most methods using primitives which explicitly have such information as part of their definition. This requires a modeller which uses bounded primitives. Using attributes to attach the geometric information to the modeller is an alternative approach and allows unlimited definition of features and their parameters. This is advantageous since, in general, the definition of features depends on the subsequent applications.

The identification of quadrics from their polynomial representation is mentioned in Chapter 8 of this thesis. This is done using eigen value and eigen vector methods and is fully described in a book by N. V. Efimov [69]

The fitting of planes to measured point data described in Chapter 8 of this thesis is done using principal component analysis. This is fully described by C. Cakir in his thesis [70].

Extracting geometric parameter information from measured data requires parameter-estimation procedures. Given that the entity type is known in advance, these procedures use numerical methods to obtain the information. V. Pratt describes a set of algorithms for fitting algebraic surfaces to point data [71]. He claims that fitting algebraic surfaces directly is more efficient than the fitting of parametric surfaces. However, for inspection purposes, it is exactly parameters, namely radii, centre coordinates, and conic angles, which are required for analysis and hence an appropriate parametric fitting is clearly more relevant.

A. Forbes in an NPL report, describes algorithms for computing least-squares best-fit geometric elements to data [72]. These procedures use a Gauss-Newton iteration approach to estimate the required parameters for planes, cylinders, cones and spheres. Chapter 8 describes how the results of these methods are used in tolerance verification.

2.11 Automatic Checking of Geometric Tolerances

One piece of recent work dealing with the determination of geometric deviations using coordinate measuring machine data is a paper written by W. Elmaraghy, H. Elmaraghy and Z. Wu [73]. The work restricts attention to geometric tolerances applied to cylindrical features and lists the various tests required in all cases. The feature parameters are estimated using a nonlinear optimization approach which is similar to the Guass-Newton method described in the NPL report (see Section 2.10).

CHAPTER 3

GENERATING THE SURFACE POINTS

3.1 Introduction

The first stage in the automatic generation of a CMM probe path is the generation of a set of points lying on the surface of the component. This requires a computer representation of the component containing information about its geometry, and an algorithm which uses such a representation to derive a set of surface points.

Solid modelling is clearly a suitable representation scheme, but no references have been found to algorithms for generating a set of points lying on the surface of solid models, the models based on either set theoretic or boundary representations.

The set theoretic solid modelling scheme used in the work described in this thesis is based on implicit half space equations. The equations may represent planar or curved surfaces. In this chapter, a method of generating a set of points on the surface of such a model is described. Two algorithms are described. The first is for faceted models, i.e. models constructed from planar half spaces only. The second algorithm is for the more complex problem of curved-surface models.

3.2 Considerations

The points being generated are for a particular inspection task and this imposes certain constraints on the positioning of the points.

Firstly, the distribution of the points over the object surface is important. If the points are too close together, time is wasted during inspection. Many points are measured within a small area, but with a limited increase in information, since the data are not representative of the whole surface. In this way, clusters of points lead to inefficient inspection.

Conversely, points too far apart will lead to an increase in the chances of missing localised defects or surface errors during inspection. Clearly, the wider the spacing between the probing points, the less representative the measured points are of the particular surface.

Finally, a distribution of points that is regular would not be ideal, either, since common types of defects due to machining are cyclic within the object surface, i.e the surface errors occur at regular spacing intervals. These errors would again be missed if the points are perfectly regular and lie on the regions of surface where there are no errors [Fig 3.1(a)]. A lobed shape is a particular example of this problem [Fig 3.1(b)].

Inspection at points which lie on edges or vertices of the object is generally to be avoided. Problems arise because probing an edge or vertex is not a well-defined procedure. When the finite sized probe hits the edge or vertex, the point of contact between the probe and the component cannot be determined. Hence the coordinates which the probe measured cannot be related to the coordinates of the point of contact, and this results in error [Fig 3.2].

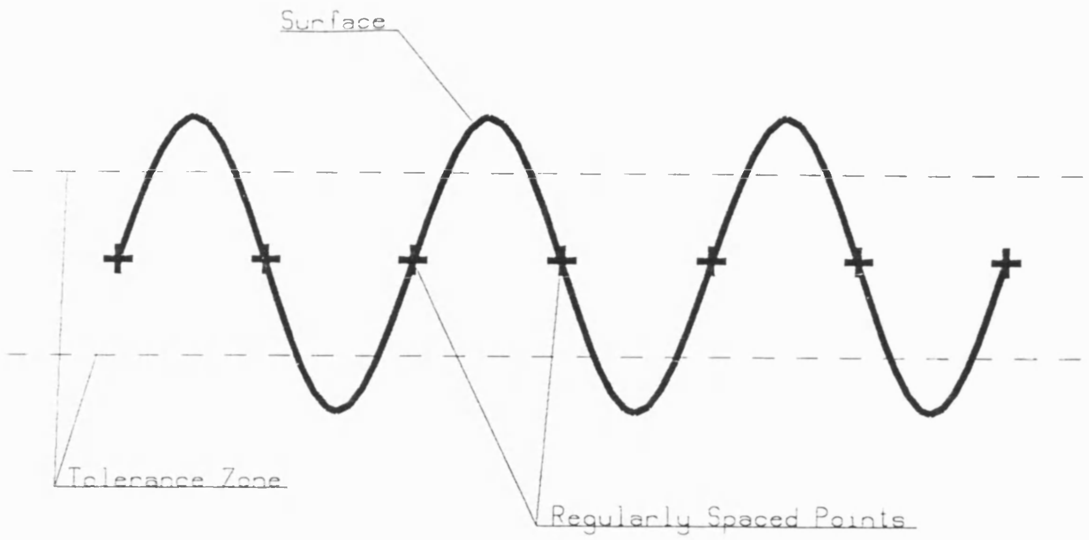


Figure 3.1(a) Missing cyclic errors.

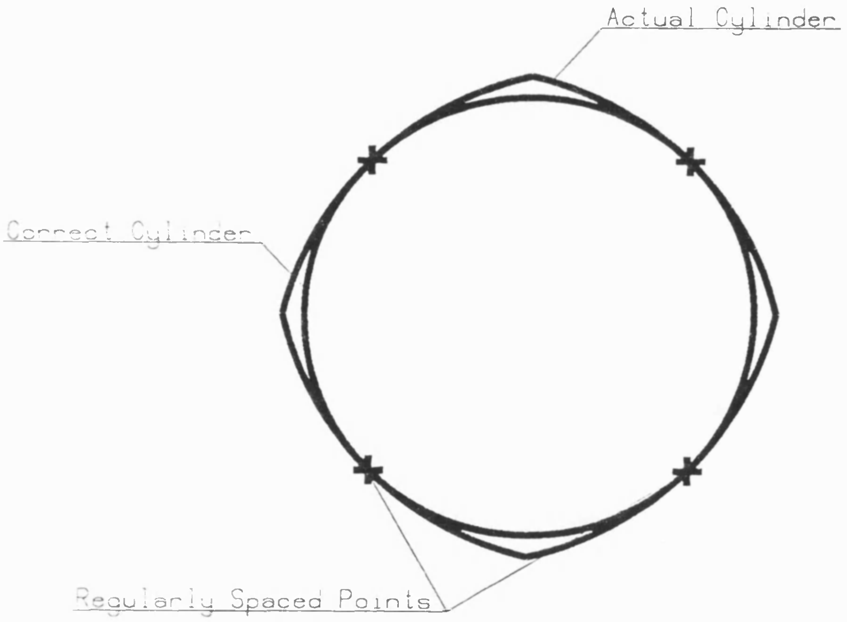


Figure 3.1(b) Missing errors of lobed shapes.

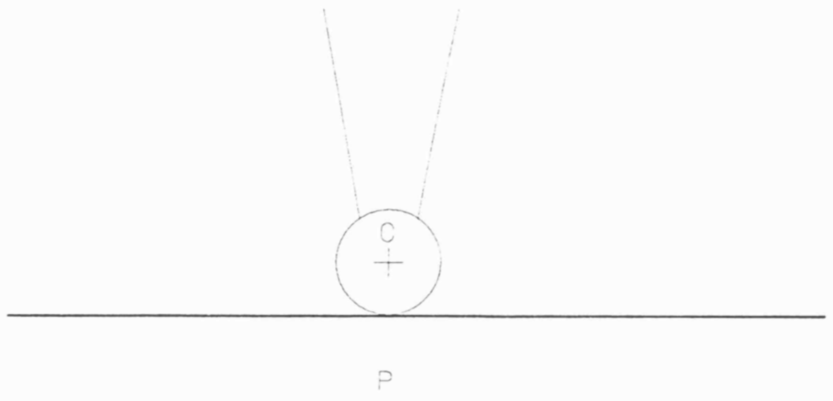


Figure 3.2(a) Correct method of probing.

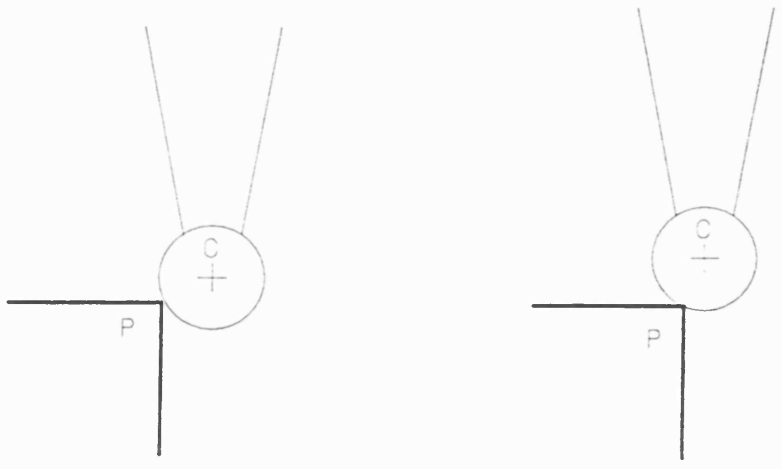


Figure 3.2(b) Ambiguous probing of edges.

Not all the geometric features of a component will require inspection, and selection of features depends on how critical their dimensions are. Further, some features may require more careful inspection than others. In view of this, only points for selected features should be generated, and the number of the points should be in some way related to the tolerancing of the feature.

The algorithm for generating points needs to be capable of yielding an acceptable point set for any object. An algorithm which is generative in the sense that it would take a set theoretic solid model as input and use this to automatically generate a set of points with the above properties would suffice. However, the solid model on its own would not allow feature selection or feature-based distributions, since these factors require additional information about the object which cannot be determined from the geometry, such as design purposes and tolerance information. In view of this, algorithms have been developed for solid models which have non-geometric information attached to them as attributes (see Chapter 7 for description of the method used to attach attributes to solid models).

3.3 Faceted Models

3.3.1 Description of Faceted Models

The set theoretic representation of objects consists of the boolean combination (i.e. set theoretic operations) of sets which represent volumes of solid. A region of space is also specified, known as the *object space*, which defines the region in which this representation is true.

With faceted models, the solid volume sets are represented by planar inequalities of the form $ax + by + cz + d \leq 0$. The coordinates of points (x,y,z) which satisfy this inequality are within solid, and the coordinates of points which satisfy the logical inverse of the inequality lie in air. The plane, therefore, divides space into two regions, one consisting of solid the other of air, and for this reason is known as a planar *half space*. Points which lie on the planar surface clearly satisfy the equation $ax + by + cz + d = 0$. Further, the planar normal direction, (a,b,c), points from the solid region into the air region.

Objects consisting solely of planar surfaces can be built using set theoretic combinations of such planar half spaces. Objects which contain cylinders and cones, may be represented to any required level of accuracy using the same kind of half spaces. For example, a cylinder can be approximated by the intersection of planes whose normal is perpendicular and pointing away from the cylinder axis [Fig 3.3(a)]. Clearly the higher the number of planes, the closer the approximation is to the actual shape.

The location of planes (within the approximation) relative to the actual surface is dependent on how the feature is to be used. For point generation, were we require points lying on the cylinder surface, it is sufficient for the points to lie close to the surface and in air. Hence the approximations would be built with the half space planes being tangential to the circumference for external features, such as the cylinder described above. In the case of an internal feature, however, such as a cylindrical hole, this approximation would give air where there is solid. An alternative approximation may then be required [Fig 3.3(b)].

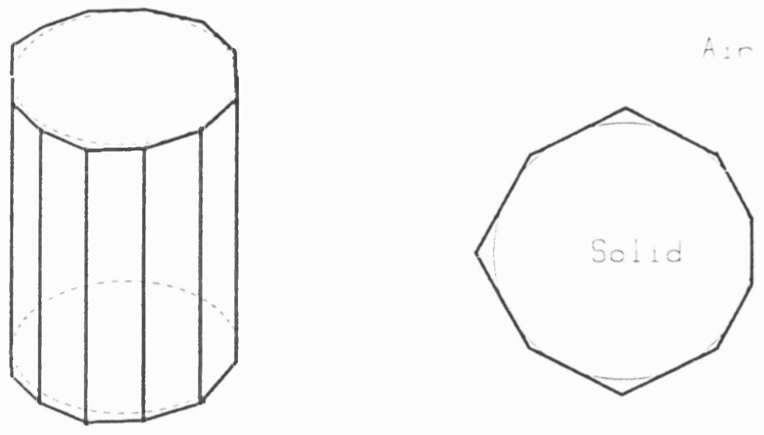


Figure 3.3(a) Facetted cylinder.

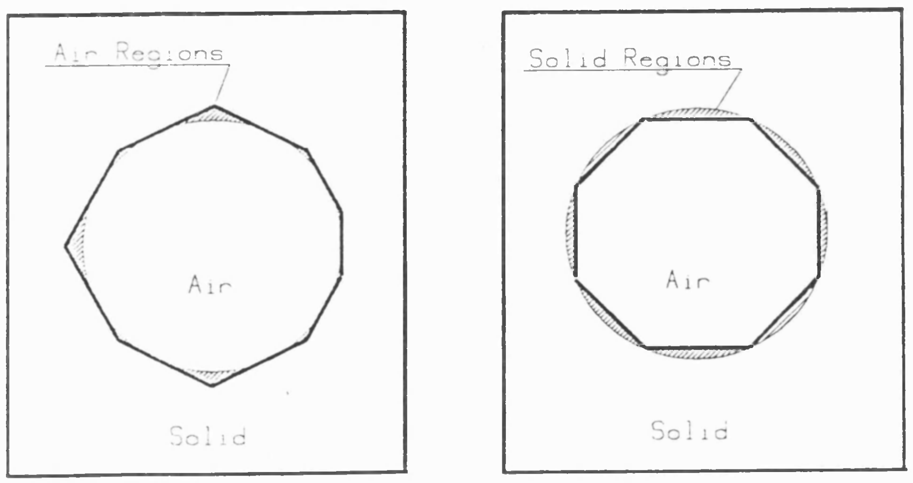


Figure 3.3(b) Alternative approximations.

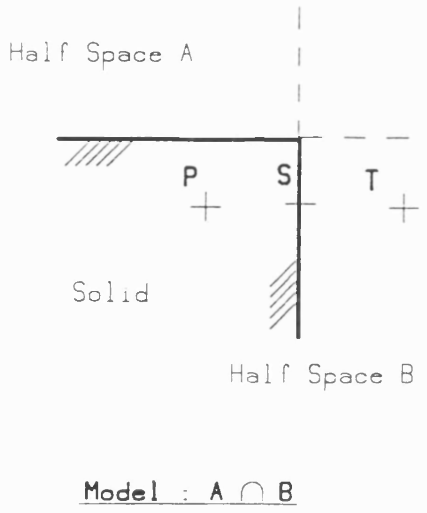
The inspection technique (described fully in Appendix 1) involves driving the probe to a point offset from the object surface, and then moving the probe slowly towards the object until it hits the surface and a measurement is taken. The generated inspection points are used to decide the position of the offset points. Once a measurement has been made, they play no further part in the inspection task, and thus the accuracy of their location is not an important factor. It follows from this that the accuracy of the faceted model is not a crucial factor in the generation of inspection points, and that the method of approximation is unimportant.

Curved half spaces can be used to model simple quadrics, such as cylinders, cones and spheres. In this representation, half spaces are not planar but second-degree polynomials in x, y and z. Clearly this offers increased accuracy, but the higher order of complexity required for consideration of these second-degree representations appears unnecessary when the accuracy of location of the points is not important.

In conclusion, for inspection of components which consist of planar, cylindrical, conic or spherical features the faceted models are adequate approximations for generating surface points.

3.3.2 Methods of Generating Surface Points on Faceted Models

Points which lie on the surface of a solid model satisfy two conditions. Firstly they lie on the surface of at least one of the constituent planar half spaces. However, not all points lying on the half spaces will lie on the surface of the model, and so secondly the points must also satisfy a surface *membership test* [Fig 3.4(a)].



For Point P :

$$\overset{A}{(SOLID)} \overset{\cap}{\cap} \overset{B}{(SOLID)} = (SOLID)$$

For Point S :

$$\overset{A}{(SOLID)} \overset{\cap}{\cap} \overset{B}{(SFCE)} = (SFCE)$$

For Point T :

$$\overset{A}{(SOLID)} \overset{\cap}{\cap} \overset{B}{(AIR)} = (AIR)$$

Figure 3.4(a) Membership testing.

∪	SOLID	AIR	SFCE
SOLID	SOLID	SOLID	SOLID
AIR	SOLID	AIR	SFCE
SFCE	SOLID	SFCE	SFCE

Union Table

∩	SOLID	AIR	SFCE
SOLID	SOLID	AIR	SFCE
AIR	AIR	AIR	AIR
SFCE	SFCE	AIR	SFCE

Intersection Table

—	SOLID	AIR	SFCE
SOLID	AIR	SOLID	SFCE
AIR	SOLID	AIR	SFCE
SFCE	SFCE	SFCE	AIR

Difference Table

Figure 3.4(b) Membership truth tables.

The test requires evaluation of the set theoretic expression defining the component, and this is done by calculating the plane equation for each half space at the test point and classifying each result as solid, surface or air. The truth tables for set theory operations as defined in solid modelling, shown in Fig 3.4(b), are then used to calculate the result of the set theoretic expression at the test point. (Precise membership testing requires consideration of *regularised* set theory operators. These extend the truth tables to consider cases where the two operands are complementary surfaces and avoid introduction infinitessimally thin sections of surface which can arise when such surfaces are combined).

The points can be chosen in various ways. For example, it is possible to make a random selection of pairs of points within the object space, with the coordinates of one point being in air and coordinates of the other being within the object. These could then be used to determine a surface point by calculating the intersection between the surface of the model and a line joining the two randomly generated points. However there is no control over the distribution of points in this method.

An alternative approach is to generate a three-dimensional grid within the object space and test the grid nodes for surface membership. However, the chances of the grid nodes lying on the object surface is small. Also, even though the points may be regularly spaced within the volume, this does not, in general, imply anything about the nature of the distribution over the object's surface. So, again, there is no control over the distribution.

The object space is a cuboid and is bounded by planes whose normal directions are parallel to the coordinate axes. These bounding planes could be used to generate points by using *ray casting* methods. A grid would be constructed on one or more of three perpendicular planes and rays cast into the object space. The points at which the rays hit the object surface become the test points. This method has the advantage of not requiring further membership tests, since the test points are already known to lie on the surface, but again allows little control over distribution. Success depends on how the object is orientated relative to the coordinate planes, and on how the grid is defined.

All methods considered so far have only used the set theoretic model description at the time of carrying out the membership tests for points. A method which also uses the surface information contained within the set theoretic expression to select the surface points would make more efficient use of the model. In view of this, an algorithm which uses the planar half space equations to generate test points has been developed. The algorithm generates a grid of test points on each half space for membership testing, and selects the successful points as the surface points. The distribution of the points over each half space is now determined by the spacing of the grid, and, although this does not guarantee control over the global object surface, it does allow local control over the distribution of points on each half space.

Because the half space may or may not contribute to the surface of the model, considering all half spaces in turn is clearly inefficient, since more membership tests will be done than are necessary. This is also true for half spaces where only a small part of the half space contributes surface to the model. The half space will be gridded throughout the object space and all grid nodes will be tested, even though only a small region contributes surface to the component.

In view of this, an algorithm which first applies spatial division and pruning techniques has been developed. Before describing the algorithm in full, the ideas and principle of object space division will be described.

3.3.3 Division and Pruning

In the set theoretic solid model representation, most of the object space will consist of solid regions within the object and air regions outside the object. The remainder will be surface. For many applications such as ray casting and surface point generation, it is this surface region which needs to be considered.

Binary spatial division of the object space involves repeatedly dividing the space into two regions and constructing two new set theoretic models which are valid for each of these two regions. In general, these new models will be simpler than the original as half spaces which do not pass through the new smaller regions can be removed from their expressions. If, after division, a half space is detected which does not pass through a given region, then the region must lie either within solid or air for that half space. Thus the half space contribution will be constant and the set theoretic expression can be simplified by replacing the half space reference with the appropriate contribution [Fig 3.5]. This makes membership testing quicker within each region.

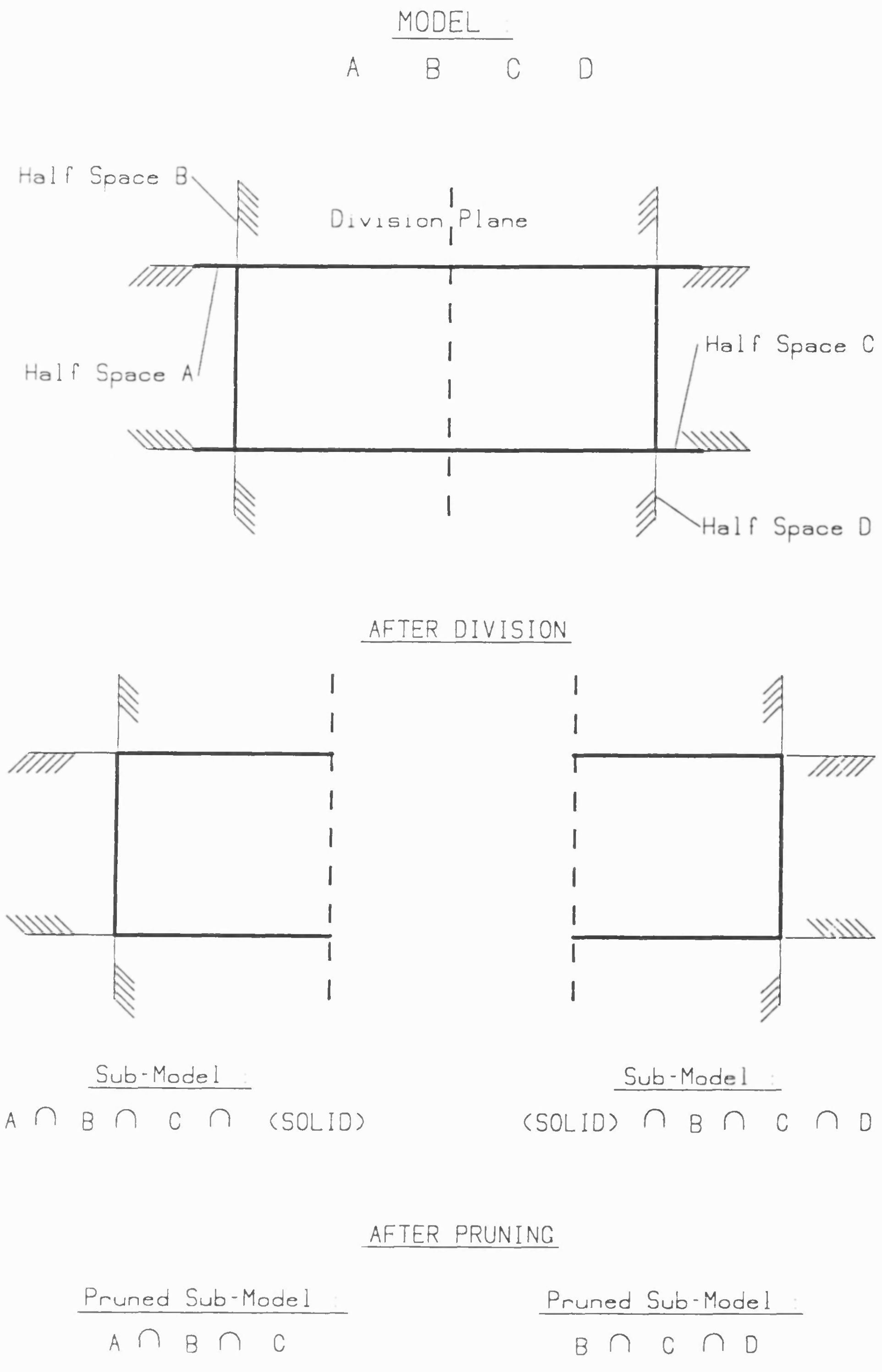


Figure 3.5 Division and pruning.

A further advantage of division is obtained by a process known as *pruning*. This uses set theoretic rules for combination with air and solid to simplify expressions further. For example, anything, either solid, surface or air, intersected with air will always yield air. Similarly, anything unioned with solid will always yield solid [Fig 3.4(b), Fig 3.5]. Using these rules, a set theoretic expression can be simplified, and, more importantly, regions consisting totally of air or solid can be detected.

Division continues within each new region until pruning results in simplification, either to air or solid, or until some division criterion is not met. This criterion is a function of complexity of the set theoretic expression and the size of the region. If the expression contains a large number of half spaces, then division is encouraged unless the size of the region is below a given level.

The decision of where to divide a region can be made in several ways. For most applications, dividing parallel to coordinate axes and half way along the largest side is simple and sufficient. When models consist entirely of planar half spaces, however, cleverer division strategies can be developed. For example, it is possible to determine zones within the regions which lie between the half spaces, and to select a zone which maximises the chances of model simplification through pruning.

The division technique generates a list of sub-space regions and associated simple set theoretic expressions which potentially contain surface.

3.3.4 An Algorithm for Faceted Models

The algorithm for generating points on the surface of faceted models begins by applying division and pruning techniques to yield a list of simpler sub-spaces and sub-models which are known to contain surface. Each sub-model is then considered in turn.

The contribution from each half space has now been isolated to a list of sub-space regions. In constructing a grid of points for each half space it is required to maintain uniformity throughout the half space. Thus a grid origin is determined for each half space and also a pair of orthogonal grid axes which lie within the planar surface. These grid parameters, the origin and the axis directions, are determined when the half space is first encountered during processing of the sub-space list.

The algorithm determines a rectangle in the plane of each half space which bounds the region contained within the sub-space. The rectangle is derived using the points of intersection between the half space and sub-space [Fig 3.6(a)].

If the half space has not been encountered before, two of the intersection points are used to determine one axis direction, and calculation of the vector product with the plane normal yields the other. The rectangle is then calculated by projecting all intersection points into these axes and selecting the extreme points along each axis. The grid origin is selected as the corner with the lowest coordinates relative to the axes [Fig 3.6(b)].

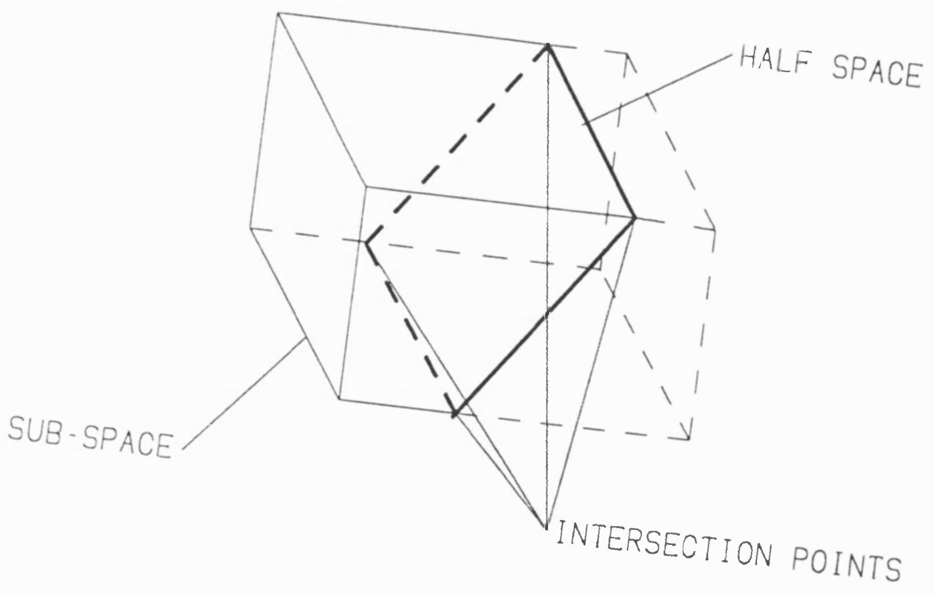


Figure 3.6(a) Half-space sub-space intersection.

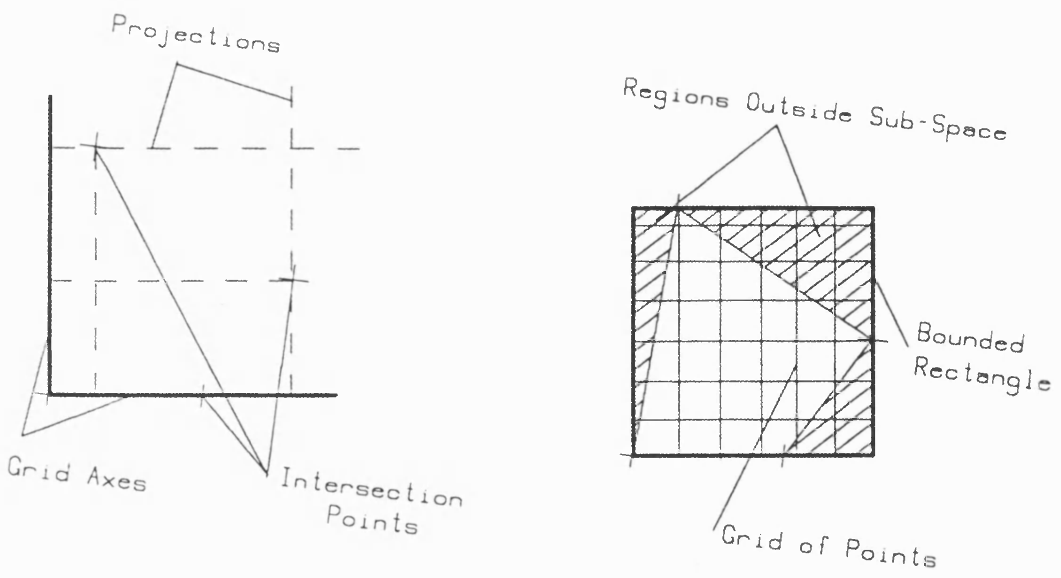


Figure 3.6(b) Construction of grid points.

If the half space has been encountered before, the axis directions and the origin are already known and the lowest grid node nearest to the intersection points is determined and used as the local grid start point. In this way, regularity is maintained throughout the half space.

Each grid node point is now tested. Firstly, they must lie within the sub-space. Since the rectangle bounds the region, this will not be true for all grid nodes. Secondly, they must lie on the object's surface, and this involves membership testing against the sub-model defined for the particular sub-space. Successful points are collected as surface points.

A regular grid is not always desirable as mentioned earlier. In view of this, the grid nodes themselves are not chosen as inspection points. Instead, a point is selected in a square which is centred at the grid node. A randomly-generated or user-defined table is used to determine a sequence of offsets from the grid nodes [Fig 3.7]. Each table entry consists of a pair of axial lengths which are used to determine a point offset from the node.

Points lying on edges and vertices can be avoided by specifying a minimum allowed distance between the points and other half spaces within the sub-model set theoretic expression. If a point is closer than this minimum distance to another half space, then it is considered to lie on its surface. A modified membership test which allows the extra operand type of "half space under consideration" and which has extra entries within the operator truth tables to accommodate the new type is used. In this way only points which lie on the particular

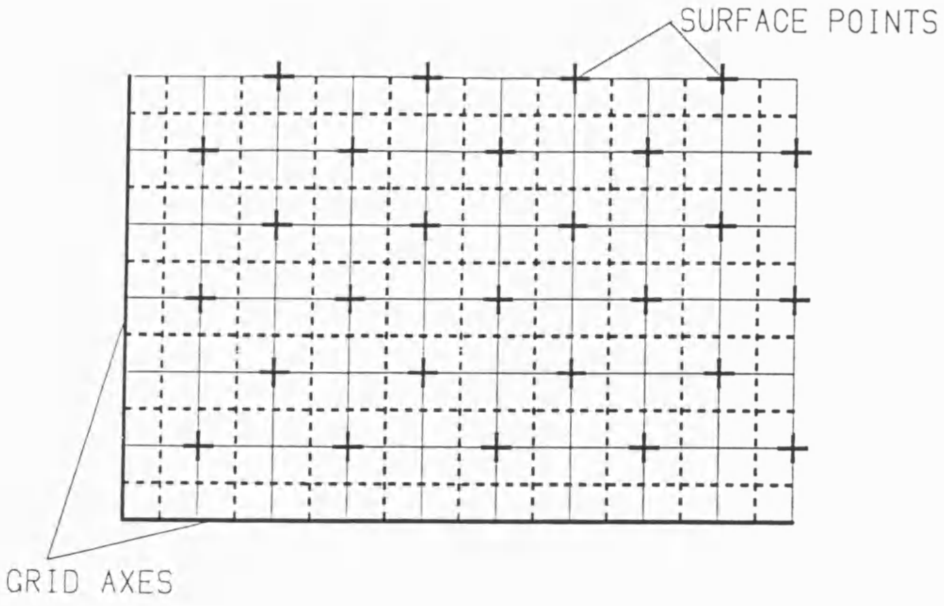


Figure 3.7(a) Points selected at grid nodes.

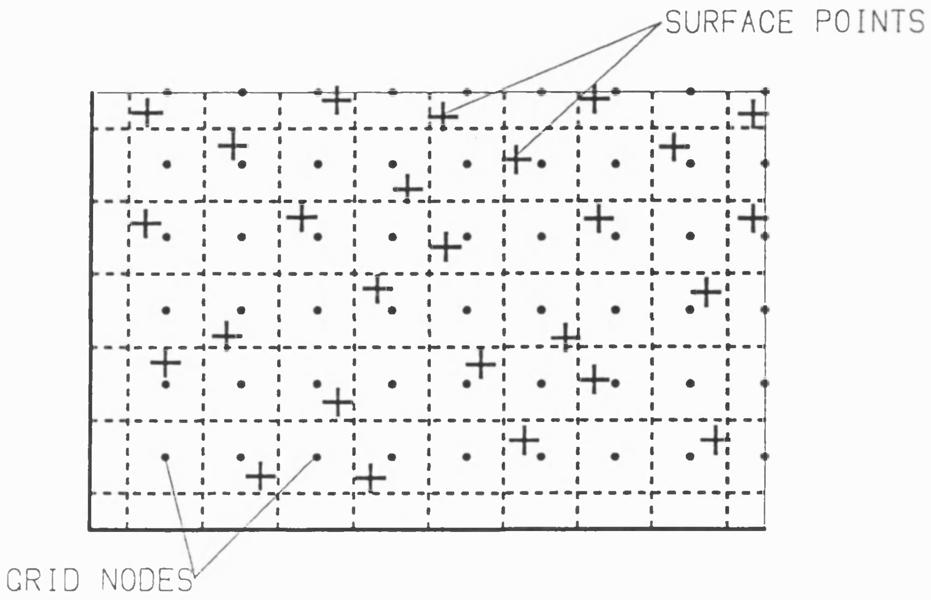


Figure 3.7(b) Points randomly displaced from nodes.

half space being considered will be chosen, since only these points will pass the membership test. (A further test should be made to handle the case where identical half spaces are referenced within the sub-model; they should clearly be treated in the same way).

However, for faceted approximations to curved surfaces, this modified test should not be used since, although edges exist within the model, they do not exist in reality and so in these cases points can lie on edges. Currently to overcome this, the modified test is only applied if the normals between the two planes differ by a significant amount, but this potentially allows edge points to be generated in non-curved faceted features. An exact method would be to use attributes to recognise when two half spaces under consideration are both part of the same faceted approximation.

Appropriate attributes are attached to the model to define features for inspection. These are applied as the model is built. Further, the grid unit size is input as a parameter, and by attaching it as an attribute of the half space, different half spaces can have different grid spacings. The choice of spacing depends on the inspection task and the area of contribution the half space makes to the object's surface.

3.3.5 Examples

The plates described here contain pictures of results of running the program on a collection of models. The pictures are wire-frame descriptions derived from the set-theoretic solid model, and the points are shown as crosses.

Plates 3.1 and 3.2 are results for a cuboid block. They illustrate the difference between using the grid nodes as surface points and using points offset from the nodes. Clearly, the first is a special case of the second with zero offsets. Attributes have been uses to select three of the six face planes.

Plate 3.3 show results for a faceted internal cylinder. Different grid spacings are used for the block and cylinder, and clearly the points on the cylinder have not avoided edges. The points lie on the grid nodes in this example.

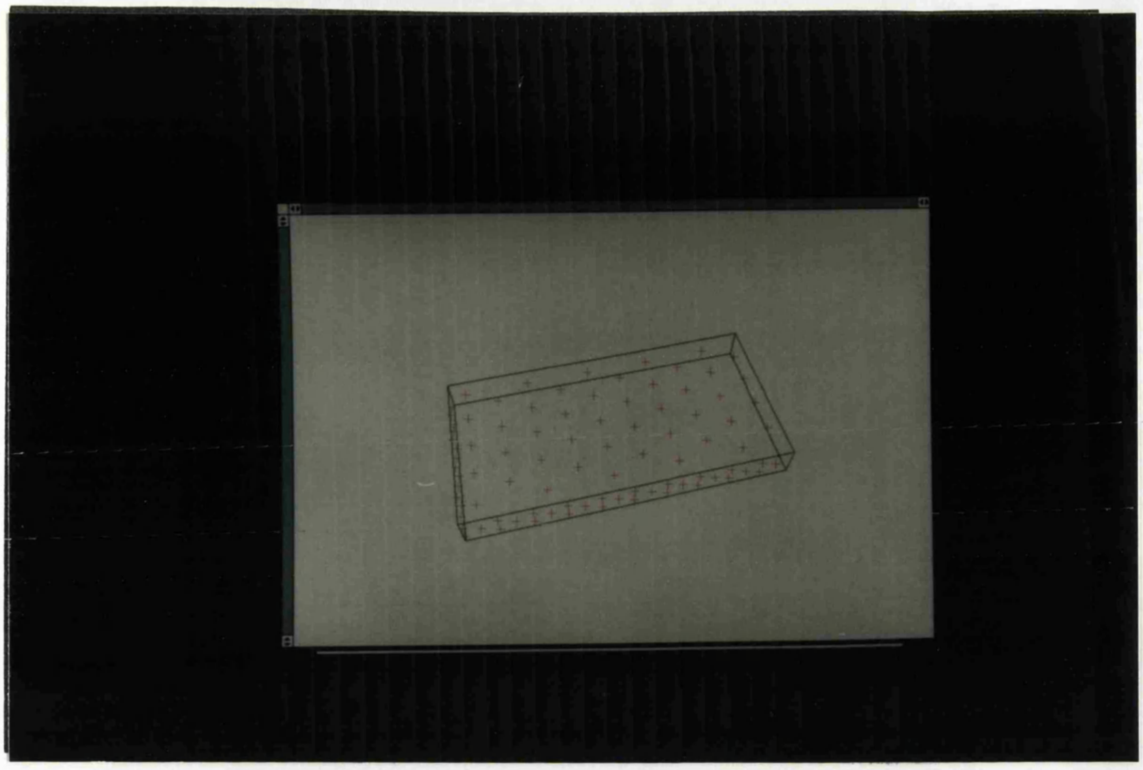


Plate 3.1 Block with Regular Grid of Points

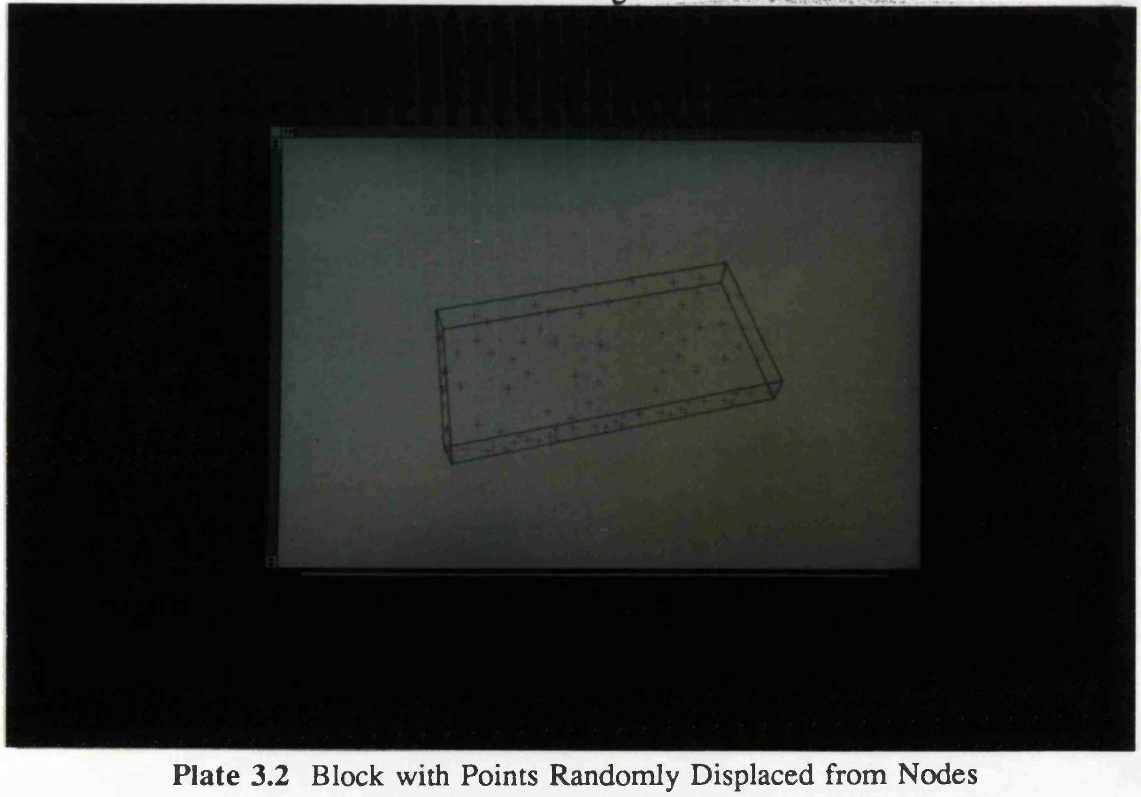


Plate 3.2 Block with Points Randomly Displaced from Nodes

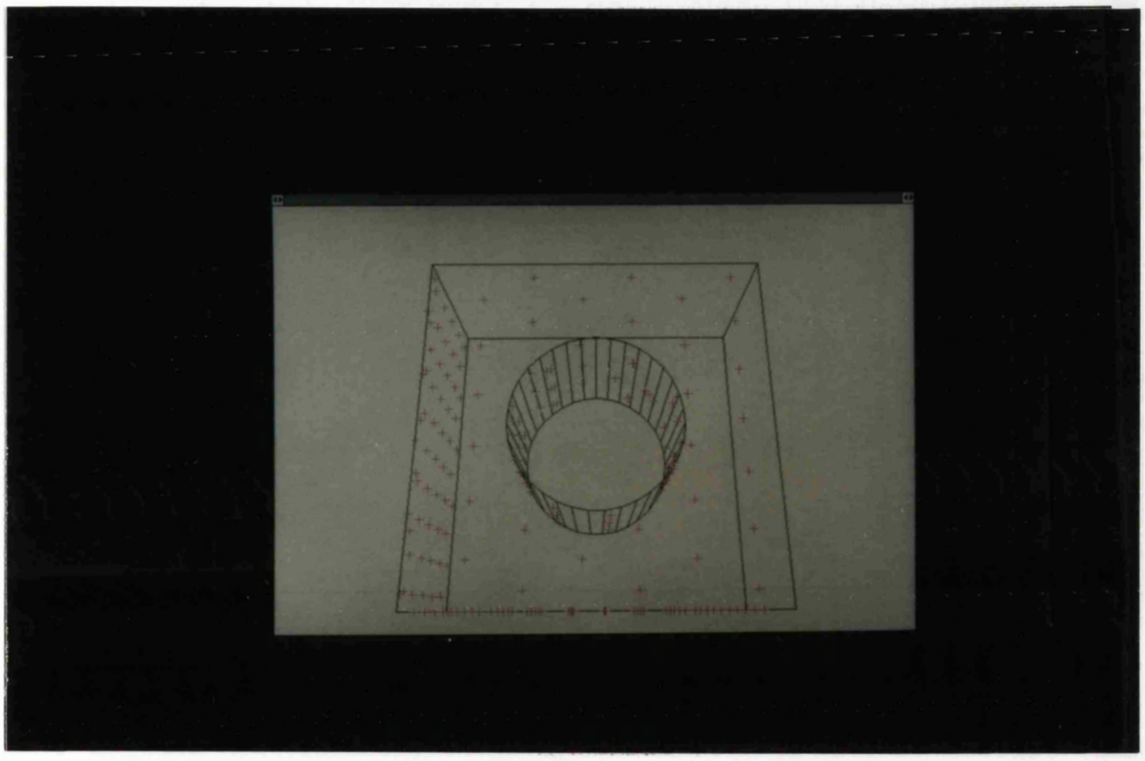


Plate 3.3 Facetted Cylinder with Regular Grid of Points

3.4 Curved Surface Models

3.4.1 Description of Curved Surface Models

Curved surface models contain set theoretic expressions between non-planar half spaces which are represented as polynomials in x, y and z. The inequality $P(x,y,z) \leq 0$ defines regions of solid. For completeness, an algorithm was developed which generates a set of surface points for curved-surface models.

Although more accurate and able to represent a larger number of shapes more easily, curved-surface models add an extra level of complexity. For example, ray casting now becomes a matter of finding roots of higher degree polynomials and surface-surface intersection tests are difficult.

The non-linearity of the surfaces makes it difficult to classify the half spaces as contributing surface, solid or air to cuboid sub-spaces. For planes, this test is easy, but for higher order polynomials it is difficult to detect whether or not a curved surface intersects a box or not. Division and pruning thus becomes more difficult and a conservative test based on the value, or *potential*, of the polynomial at the sub-space corners is used to decide if the polynomial intersects the box or not.

The problem of generating surface points again involves considering a list of sub-models which may contain surface.

3.4.2 Extending the Faceted Algorithm

The faceted algorithm relied on the linearity of the planar half spaces in order to generate a grid of points. With curved surface models, the problems associated with gridding the half spaces are considerable.

Firstly, determining an orthogonal pair of axes for the grid involves determining a set of curves lying in the surface which are perpendicular where they intersect. Such curves can be obtained by calculating the intersection curves between orthogonal planes and the surface. Thus, the sub-space is divided into a grid of planes and the curves corresponding to the intersection of these planes with the curved half space form the surface grid.

However, it is difficult to determine whether these curves of intersection have a real range. Geometrically, this is related to the problem of determining whether the plane intersects the surface. For simple cases, such as, for example, a sphere, this may be possible to detect, but for general polynomials this is not easy with algebraic representations of surfaces. Further complications arise when closed curves occur, since termination conditions have to be found.

Secondly, selecting equally spaced points along a given curve involves solving a integral equation. Again this is difficult and requires the use of numerical techniques.

In view of these problems, an alternative algorithm has been developed which uses techniques from interval arithmetic. The principles and techniques of interval arithmetic are fully discussed by R. Moore (see Section 2.5.2 for references).

3.4.3 An Algorithm for Curved Surface Models

As with the algorithm for faceted models, the algorithm for curved half spaces generates a list of sub-spaces containing half spaces. A plane is then determined which faces most of the half space segment lying within the sub-space and this plane is used to determine a set of surface points. The plane is, in some sense, the most tangential to the surface in the sub-space.

For each half space, the polynomial equation of the normal is obtained using partial differentiation. For a majority of widely used features, such as cones, cylinders and spheres, these derivatives are linear. Ideally, the mean normal over the sub-space is now required, and this involves calculating the integral

$$\underline{\hat{n}} = \frac{\int_{A_s} (P_x, P_y, P_z)\, dS}{\int_{A_s} dS}$$

where A_s is the surface region lying within the subspace, $P(x,y,z)$ is the equation of the surface and P_x, P_y and P_z are partial derivatives of P. However, such an integral is difficult to compute since the area A_s is not readily available, and a less accurate but simpler method which uses interval arithmetic is used.

Interval arithmetic is used to determine the range of direction of the surface normals within this sub-space by evaluating the partial derivative polynomials for the intervals which define the sub-space. The midpoint of the resulting intervals are then chosen as the plane normal direction. This bears little resemblance to the integral above, which is weighted by the distribution of the normals rather than just their range, but the estimate suffices for the cases where a sub-space contains parts of cylinders, cones and spheres.

Once the direction has been determined, a plane is determined outside the sub-space with a normal direction pointing in. A grid is constructed lying within this plane in the same way as with faceted models, being bounded by the projections of the sub-space corners into the plane. Surface points are generated by casting rays from the grid nodes along the planar normal, and determining the intersection points with the sub-model [Fig 3.8].

If the sub-space within which the sub-model is defined is sufficiently small, then the surface of all half spaces passing through the sub-space will be locally flat (i.e. almost planar within the sub-space). If a plane is determined as described above for a half space within this sub-space, then normal direction of this plane will be parallel to most of the normal directions of the half space within the sub-space. Hence, a majority of the rays cast from this plane along its normal direction will intersect the half space, and hence generate a tentative surface point.

This method allows no control over distribution since a new plane has to be calculated for each intersection between sub-spaces and half spaces. Edges and vertices are still avoided by not selecting points which are close to the zeroes of two or more curved surfaces. Attributes are used to define inspection features and grid sizes. Control of the distribution is left totally to the post processing-algorithms described in Chapter 5.

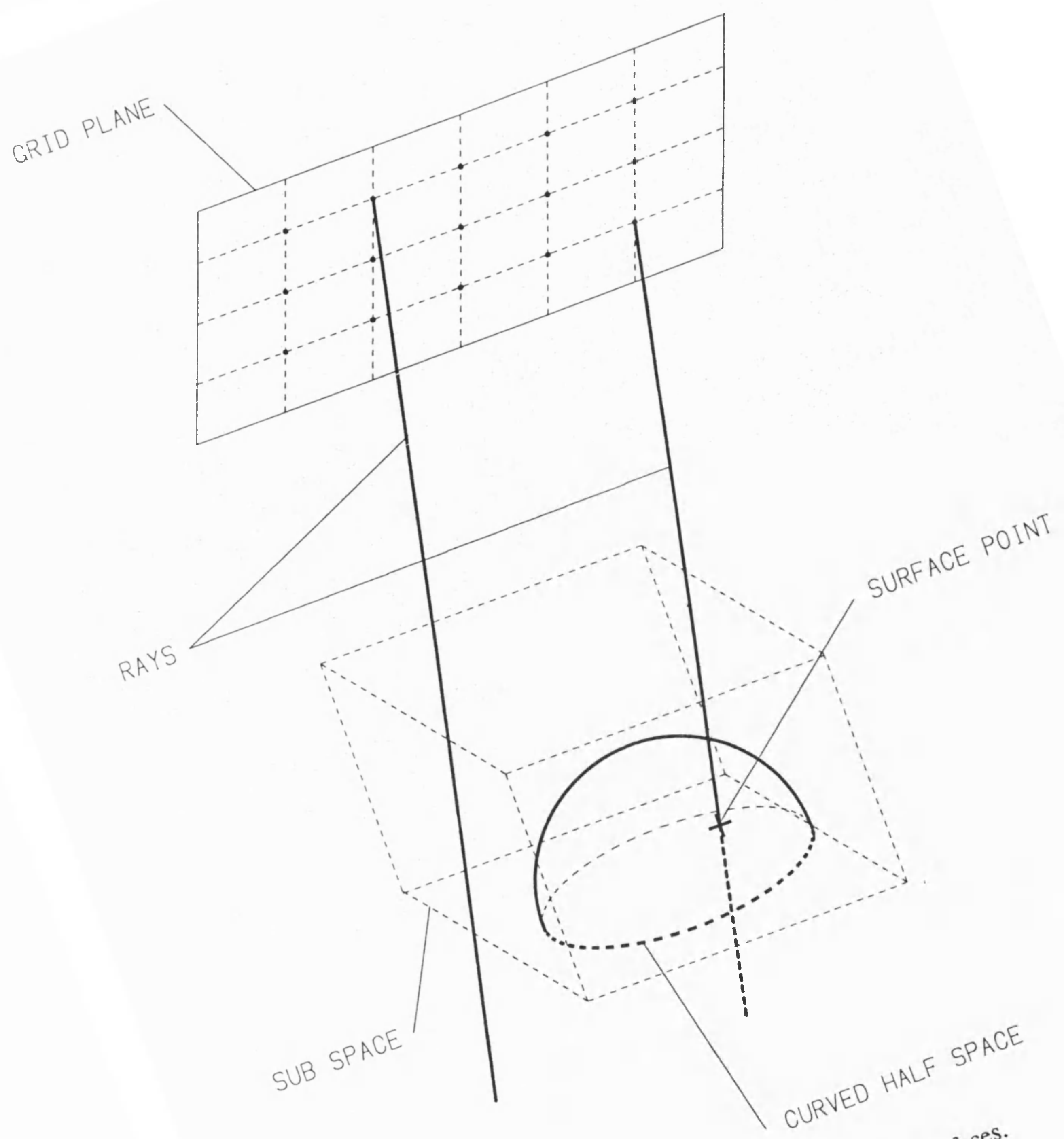


Figure 3.8 Generating points on curved surfaces.

3.4.4 Examples

The algorithm has been successively used on common quadric features, such as cylinders and spheres, and on objects containing these features. The pictures illustrate results.

Plate 3.4 displays the results for a single cylinder. The picture was generated by ray-casting into the set theoretic solid model and the line segments represent the points. The lines are orientated at a direction normal to the surface. Attributes are attached to the model to select only the curved surface features for surface point generation.

Plate 3.5 shows a feature comprising of a mixture of cylindrical and faceted features. Attributes were used to give different grid units to the various features.

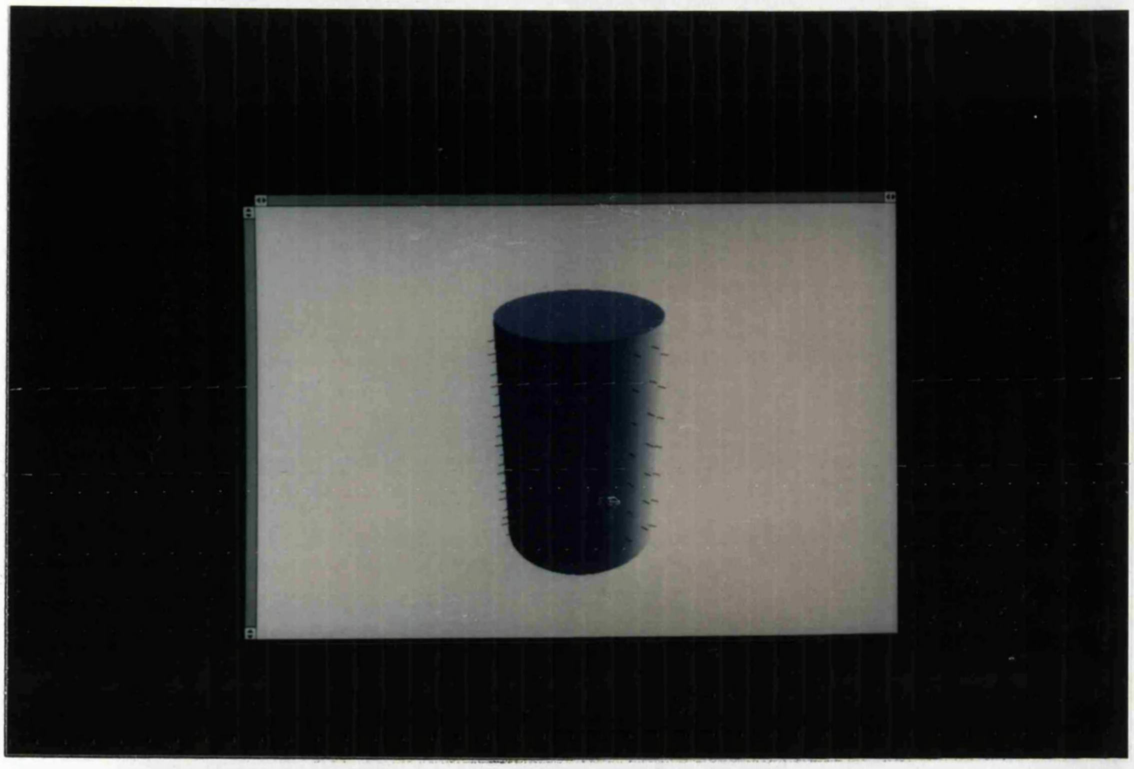


Plate 3.4 Cylindrical Surface with Surface Points

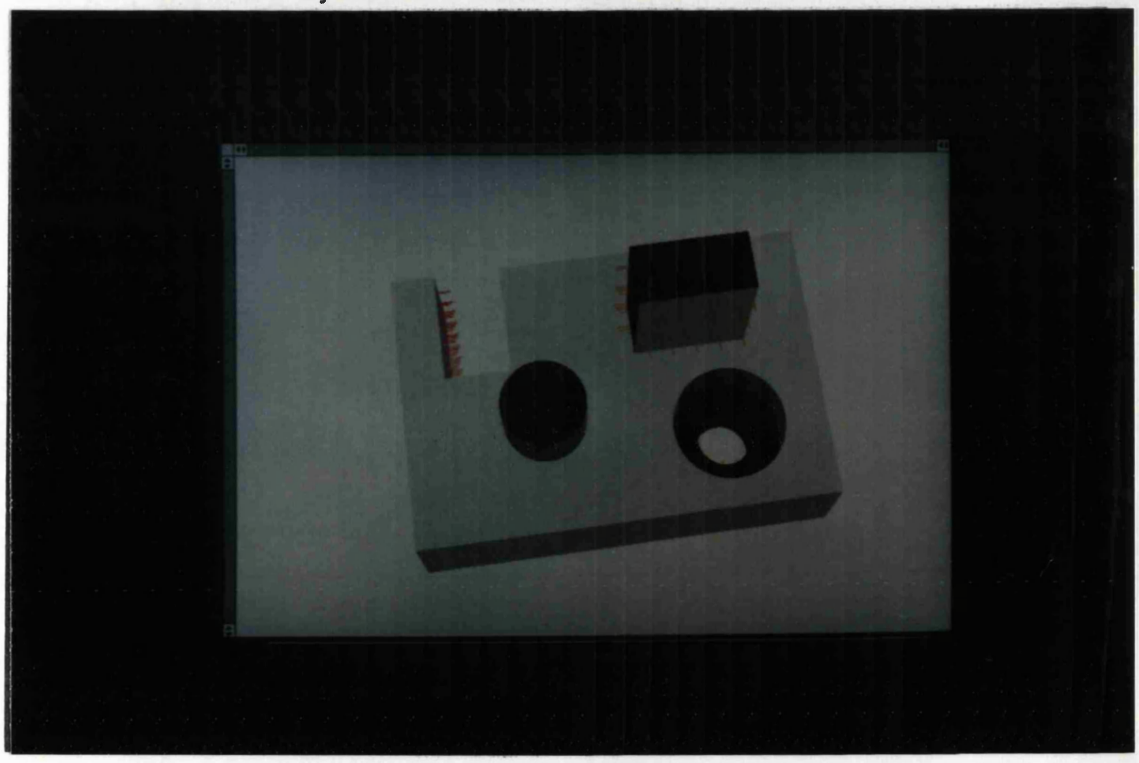


Plate 3.5 A More Complicated Curved Surface Example

3.5 Other Methods of Generating Points

From the previous sections, it is clear that finding a satisfactory method which allows total control of point distribution is difficult. For common features, such as those for which geometrical tolerances are defined, attributes may allow better control by allowing more information to be transferred to the point generation program. For example, if a set theoretic expression is known to represent a cylinder, this information together with appropriate parameters may be useful in determining inspection points. British Standard BS7172 [section 2.7] offers appropriate guidelines, and these could be used to calculate a set of points for one size of each feature. These points can then be transformed to fit any size.

The problem with this technique is that although, for example, cylinders, cones and spheres, are used in the definition of the model, they may not contribute significantly to the surface of the resulting model. As an example, consider the component obtained when one cylinder, A, is differenced from another, B, whose axis is orthogonal to A. All points lying on the surface common to both A and B will not lie on the component's surface.

For curved surface models, an alternative is to generate points on a faceted version and use these as ray casting nodes to obtain the real points. For simple quadrics, and with the appropriate attribute information, it is an easy matter to automatically facet the features. For higher order surfaces, however, it is not clear how to automatically construct faceted approximations other than by hand.

3.6 Conclusions

The method of generating surface points on faceted models described here fulfills all considerations of inspection point generation, except that there is limited control on distribution of points. Edges and vertices are avoided, and simple attributes, which may be geometrical tolerances, allow selection of features with varying grid unit sizes.

Since the distribution of points is a global property of the set of points, it is unreasonable to expect a point generation algorithm to be able to control fully the distribution and fulfill all considerations described in Section 3.2 above. To do so would add an extra level of complexity since the distribution of points would need to be recalculated every time a new point is added. A much better approach is to modify the set of points after generation as a post process step when all the necessary information is available. This involves eliminating points in order to make the distribution better, and thus requires that a sufficient number of points have been generated initially. Chapter 5 describes the selection of *inspection* points from the set of surface points.

If a feature has been tagged for inspection, but the point generation algorithm failed to yield a set of points, a smaller grid size is needed. In principle, the program can iteratively find the grid size which first yields an appropriate number of points. Such an algorithm may be slow, depending on the original grid unit size and the contribution the feature makes to the surface, and perhaps a better method would be to derive a grid size from an estimate of the surface area of the feature. In either case, it is clear that automatically estimating the grid size is a feasible option.

CHAPTER 4

GROWING SET THEORETIC SOLID MODELS

4.1 Introduction

Having generated a set of surface points for a component, the next stage is to select a set of inspection points. Inspection points have to be accessible to the CMM probe, and thus the probe should be able to reach them without colliding with any other part of the component. Further, when measuring an object, unexpected collisions between the probe and the object must be avoided. To overcome these problems a means of detecting collisions at the inspection planning stage is required.

Ray casting along straight line path segments into the model will detect collisions which would occur when a point moves along the path segment. Although this method would detect obstructions along path segments, successful passage for a point does not always imply safe passage for probes which have finite volumes. An improved ray casting approach to the problem of collision detection has been developed which is based on increasing the volume of solid occupied by the model. By appropriately enlarging the model, the probe can be treated as a point relative to this expanded model and normal ray casting techniques can be applied. (As mentioned in Chapter 2, this is related to the configuration space approach described by T. Lozano-Perez. This chapter describes how the configuration space obstacle for a CMM probe can be approximated using a set theoretic model).

A *grown* set theoretic model is defined as a set of points whose minimum distance from the original model is less than a growth parameter, d. The concepts and techniques of growing set theoretic solid models are described in this chapter, followed by a description of how the techniques can be applied to collision detection for CMM measuring probes.

4.2 The Concept of Growing

The effect of growing a set theoretic model is to displace the surface boundary along its outward normal. This is not the same thing as increasing the model size by scaling, which would maintain the relative positional relationships between features. For example, growing a model which contained a cylindrical hole would produce a new model in which the radius of the cylinder is smaller. Scaling the same model would produce a model in which the cylinder will have a larger radius. In general, growing has the result of expanding external features and contracting internal features [Fig 4.1].

The set theoretic solid model definition consists of the set theoretic combinations of half space primitives. Each primitive may or may not contribute to the surface of the model. A new model is constructed by maintaining the same set theoretic expression, and changing the half spaces. These new half spaces are chosen to be offset from the original half spaces in a direction parallel to the surface normals. If difference operators have been re-written in terms of intersections and half space complements, then this new model will contain surface segments offset from the original surfaces.

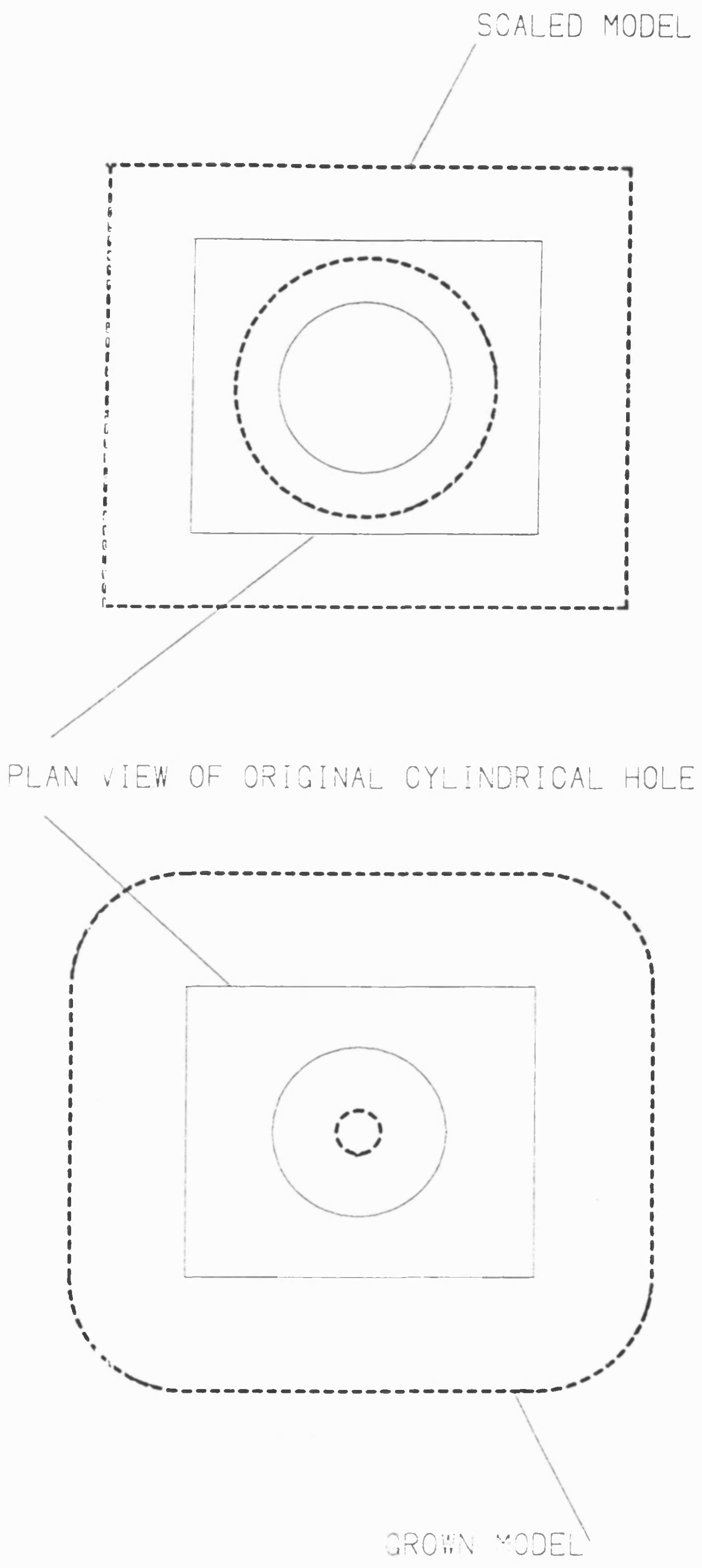
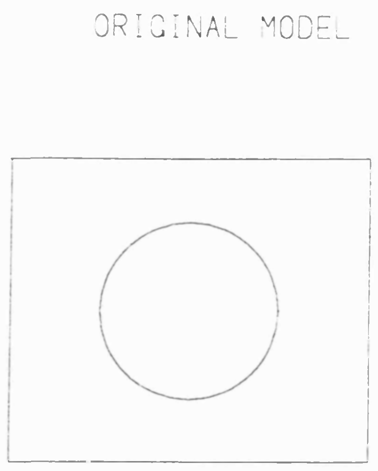


Figure 4.1 Difference between Growing and Scaling.

The above can be readily applied to faceted models where the change to half spaces involves shifting planes. However, the new model constructed in this way is only an approximation of the grown model. This becomes clear when the behaviour at edges and vertices are considered. Fig 4.1 shows the correct grown model (obtained by applying the definition given in Section 4.1) which results when growing a component consisting of a square block with a vertical cylindrical hole. The corners of the block have become rounded corners i.e. cylindrical surfaces, and this will be the case for all edges. In the same way, vertices will become spherical surfaces in the grown model. In contrast, the model obtained by shifting the planar faces will contain edges similar to those in the scaled model as shown in Fig 4.2. The approximation to the grown model will contain the correctly grown model as a sub-set, as Fig 4.2 shows, and so gives a conservative estimation which is adequate for collision detection.

For simple curved half-spaces entities, such as cones, cylinders and spheres, it is possible to calculate a grown entity given the parameters which define the original entity, (i.e. the axial direction, radii etc). However, if the half spaces are curved, it is not clear how to derive a new half space which is offset from the original half space. Indeed, the only successful way seems to be to fit a surface equation through a set of offset points. Further, the behaviour at edges and vertices of intersections between curved surfaces becomes much more complicated. However curved surfaces can be approximated to an arbitrary degree by facetted models and so restricting attention to facetted models does not unduly restrict the possible applications of the developed techniques.



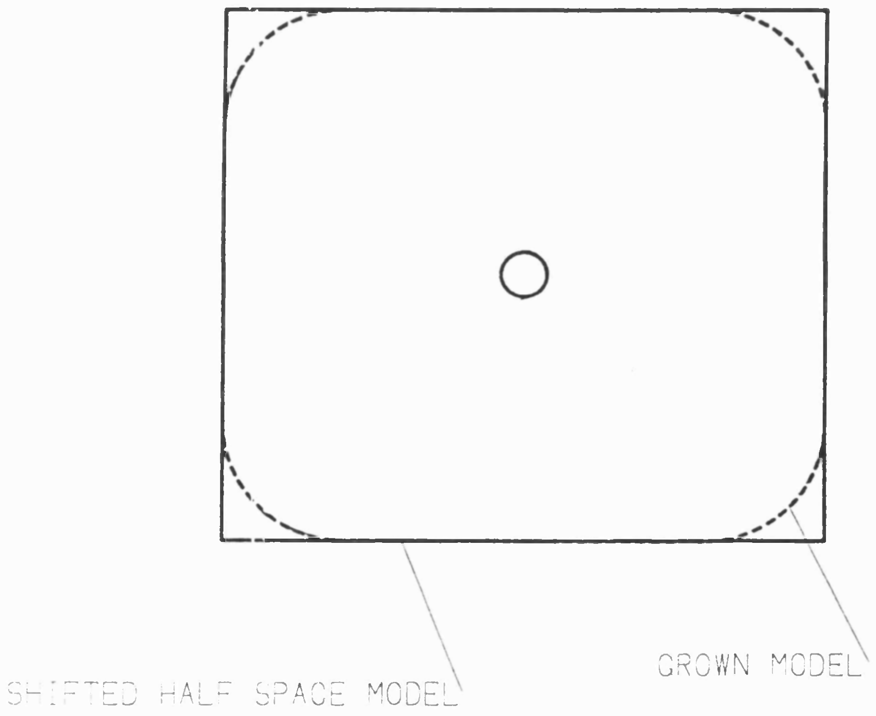


Figure 4.2 Difference between Growing and Shifting half spaces.

A half space planar equation has the form:

$$P(x,y,z) = ax + by + cz + d < 0$$

To shift the half space a distance λ along its normal requires the shift operator $S_\lambda(P)$ defined as:

$$S_\lambda(P) = P - \lambda$$

Thus to obtain an approximation to a grown facetted model, $S_\lambda(P)$ is applied to all half spaces $P(x,y,z)$ within the model.

The new model obtained by shifting half spaces may have the same topology as the original model. However, this is not always the case. Because of the global nature of the half spaces, it is difficult to determine or control the changes which occur in the complete model when a half space is moved. For example, if one half space passes through another during the growing process, a half space which did not contribute surface in the original model may now do so in the grown model [Fig 4.3]. This could result in unwanted topology changes within the grown model.

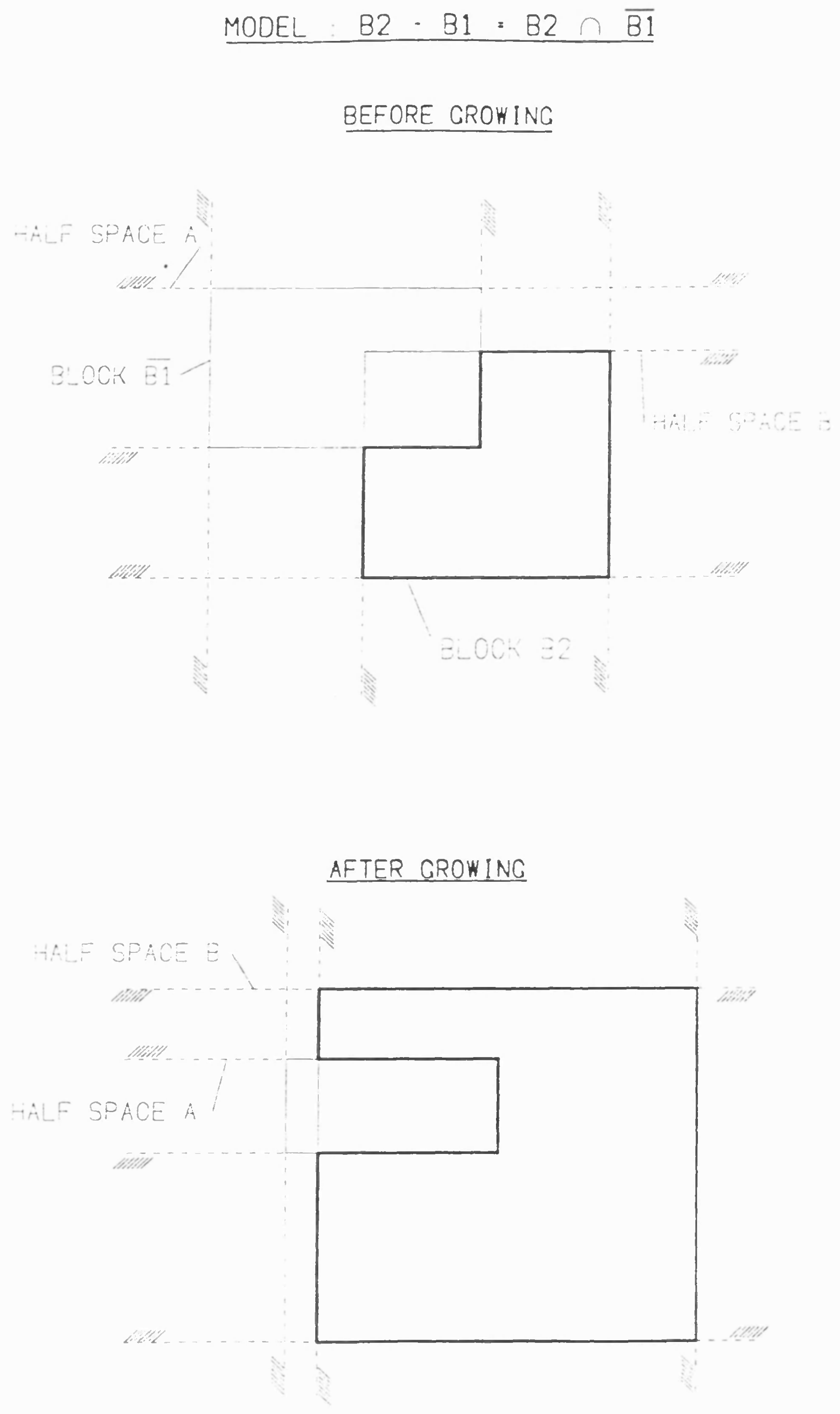


Figure 4.3 Topology changes due to Half space A passing through Half space B

Not all topology changes are unwanted. For example, when a component contains internal features, such as slots and holes, growing of the model results in the contraction of such features. If the growing involves shifting surfaces a sufficiently large distance, internal features may disappear from the grown model.

Another example of topology change can occur when features are unioned together. For example, if a cylinder is unioned with a block, new edges may appear at the boundary as the block face grows into the cylinder and the cylinder position is unchanged [Fig 4.4]. This kind of topology change is not of concern for the application of growing described in this thesis (namely, collision detection) since the resulting model encloses the expected grown model and so is a conservative approximation.

Intuitively, all objects can be grown in a sensible and consistent way without unwanted topology changes, and in practice this appears to be true for a majority of cases. Topology changes occur in facetted models when new vertices appear in the grown model, or when vertices which exist in the original model no longer exist in the grown model. This occurs when a half space passes through an intersection point of three or more planes, i.e. when the relative position of such a point to a planar half space changes. Section 4.4.1 describes the reasons for this in more detail; it is sufficient here to say that when such events occur, there is a greater chance of changing the vertices of the model.

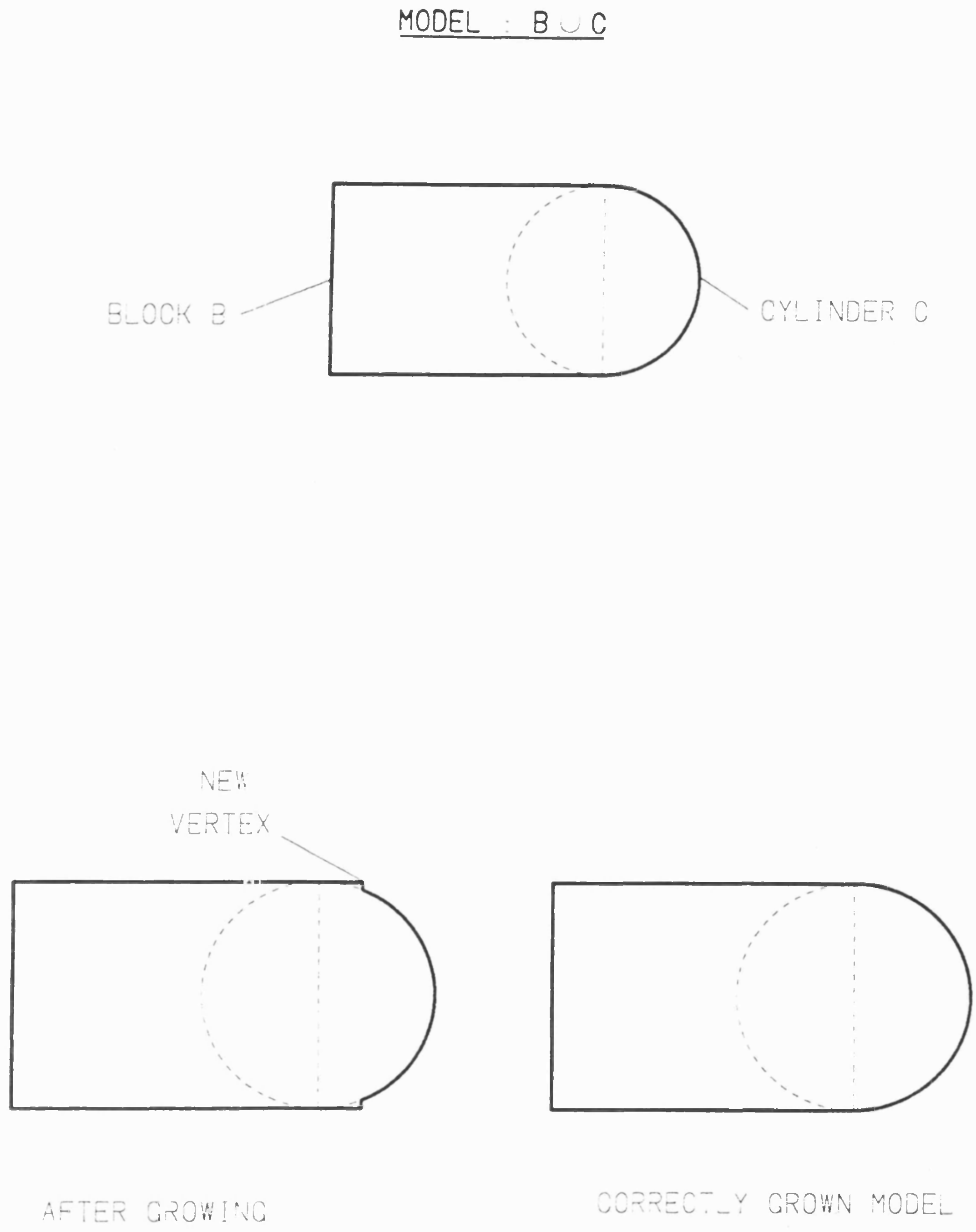


Figure 4.4 Topology changes during growing of cylinder unioned with block.

4.3 Set Theoretic Tree Reconstruction and Growing

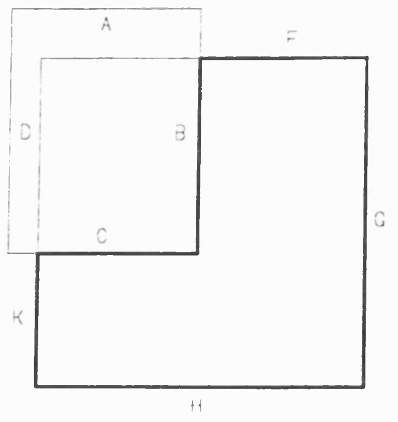
The set theoretic expressions representing any model may be rewritten with differences redefined in terms of intersection and complement operators. For example, in differencing planar half space represented by the implicit equation $Q(x,y,z)$ from planar half space represented by $P(x,y,z)$, the following rule is used:

$$P - Q = P \cap \overline{Q}$$

where $\overline{Q}$ is the complement of Q. Planar half space complements are constructed by negating the coefficients of the half space, namely a, b, c and d.

Consider a set theoretic expression consisting of half spaces and boolean operations between them. Difference operators can be removed from the expression by rewriting them in terms of intersections of complements, and this results in a new expression. By *distributing* intersections over unions in this new expression, an expression consisting of the union of *intersect-models* can be obtained, with these intersect-models consisting solely of intersections between half spaces [Fig 4.5]. (This is directly analogous to the distribution of multiplication over addition to yield the sum-of-product form).

MODEL : B1 — B2



BLOCK B1 $\quad F \cap G \cap H \cap K$

BLOCK B2 $\quad A \cap B \cap C \cap D$

$$B1 - B2 = B1 \cap \overline{B2}$$
$$= (F \cap G \cap H \cap K) \cap (\bar{A} \cup \bar{B} \cup \bar{C} \cup \bar{D})$$

DISTRIBUTED MODEL

$$B1 - B2 = (F \cap G \cap H \cap K \cap \bar{A}) \cup (F \cap G \cap H \cap K \cap \bar{B}) \cup (F \cap G \cap H \cap K \cap \bar{C}) \cup (F \cap G \cap H \cap K \cap \bar{D})$$

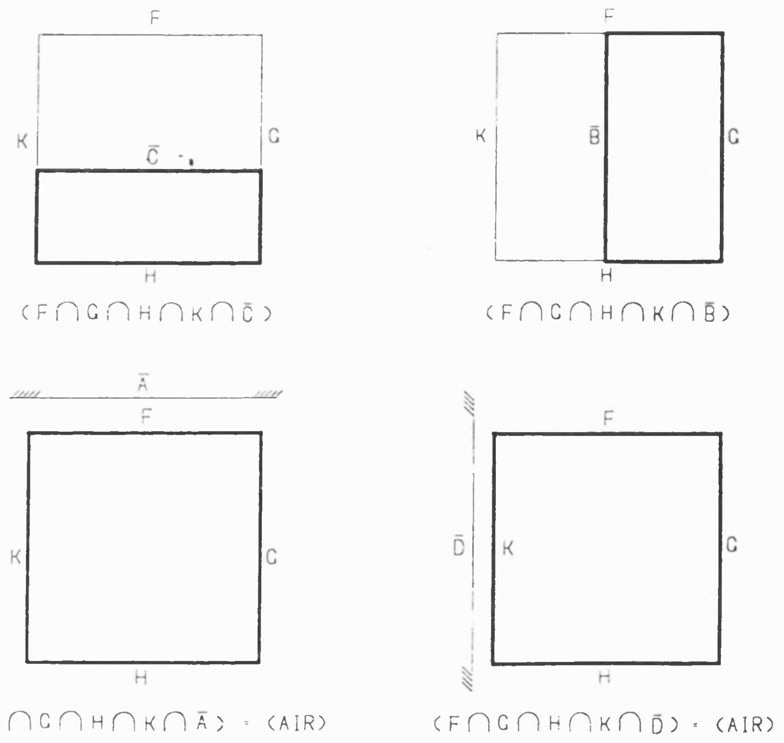


Figure 4.5 Tree Reconstruction. Some intersect-models can be pruned to air.

If an intersect-model is non null then, since it contains only intersections between half spaces, it is convex. Further, the cases where an intersecting point between three or more planar half spaces, (i.e. a tentative vertex), within the intersect-model can change its position relative to any other half space within the same intersect-model will result in topology changes which always give conservative estimates to the exact grown sub-model. To see, consider when there are four half spaces which intersect at a point, T. When the half spaces are shifted a distance λ along their normals, the intersection point T will be translated a distance which is greater than the shifted distance λ and hence will pass through half spaces [Fig 4.6(a)]. The effect of this is to introduce three new vertices. However, this case is not a problem because the resulting grown intersect-model is still a conservative estimate of the exact grown intersect-model [Fig 4.6(b)]. (In fact, the case is similar to the example already discussed in Section 4.2 where a block and cylinder are unioned together).

If the intersect-model is null it is quite possible that shifting half spaces will result in a non-null intersect-model as half spaces which face each other pass through each other and introduce new solid regions. This in turn will lead to topology changes in the total model since new contributions of solids are unioned into it. If the null intersect-models are detected, these can be trivially removed from the model just by removing an intersect-model and a union operator pair from the distributed model expression. This can be done because the union between null and any model M, say, always results in M. Of course, null object detection is non-trivial as described in chapter 2, but algorithms may be able to exploit the fact that each intersect-model consists only of intersections between planar half spaces. For example, half spaces facing each other can be detected, and intersect-models consisting such pairs of half spaces are null.

Another problem arises with *redundant* intersect-models. These are non-null intersect-models which are completely contained within other intersect-models, and thus whose contribution to the model is not needed. When half spaces are shifted to grow the model, it is possible for a half space within this intersect-model to pass through a half space intersection point of the intersect-model which contains it. As Fig 4.7 shows, this results in topology changes as new vertices appear. As with null intersect-models, redundant intersect-models can be trivially removed from the model although their detection is not simple.

If the null and redundant intersect-models are removed, the resulting model now consists of the union of a list of convex sub-models. Growing each sub-model will result in a new grown intersect-model as described above. The union of these intersect-models will then result in the grown original model.

Null object detection aside, another problem with this approach is the combinatorial increase in the number of half spaces which occurs during distribution. Limited simplification is possible. Repeated half spaces within sub-models can be removed. However, the resulting models are, in general, very large [Table 4.1].

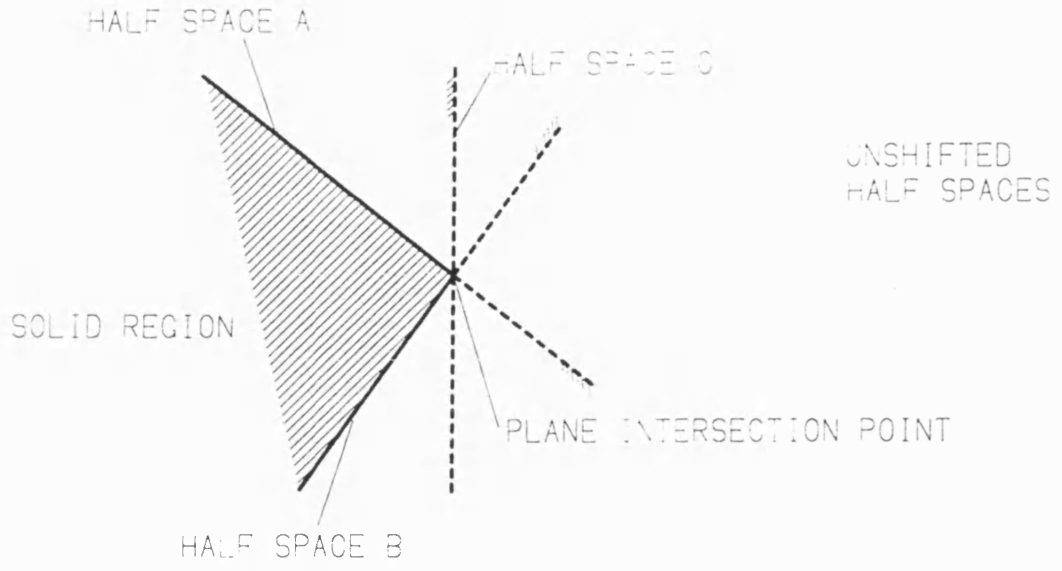


Figure 4.6(a) Topology changes with three or more half-spaces.

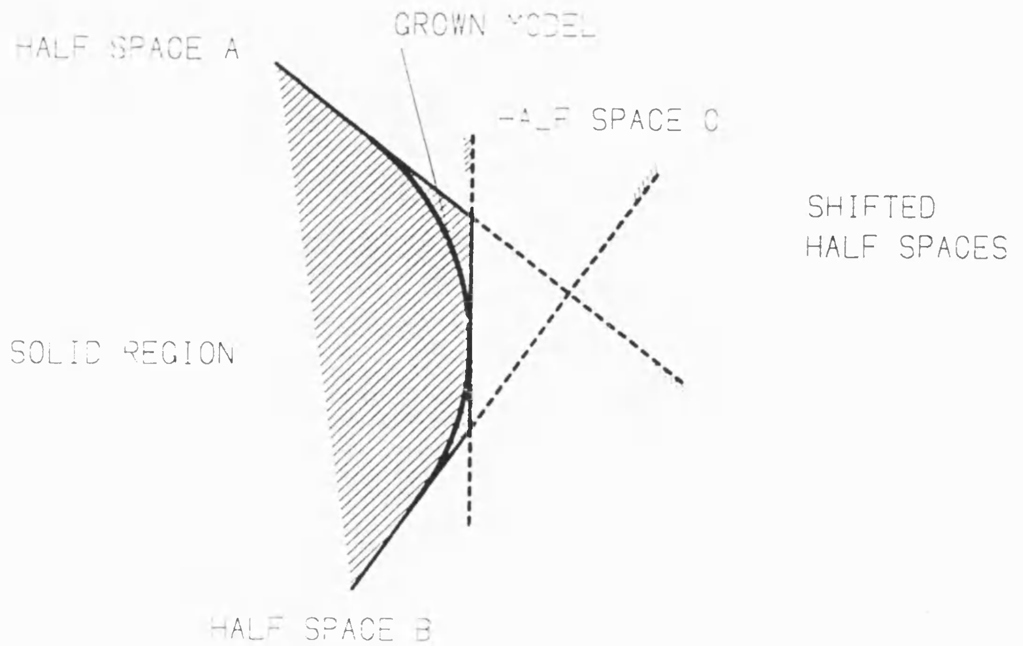
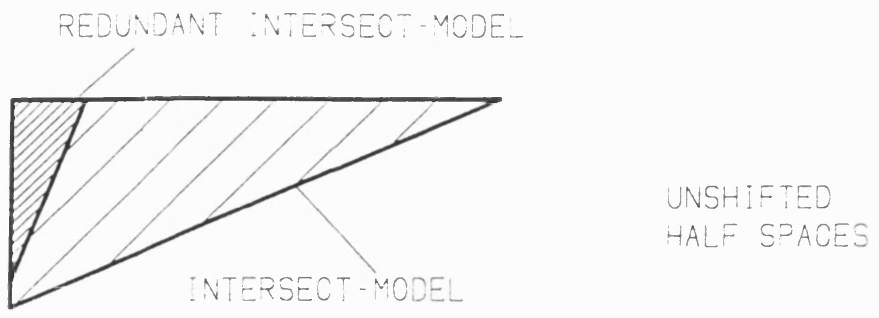


Figure 4.6(b) The changes produce a conservative model.



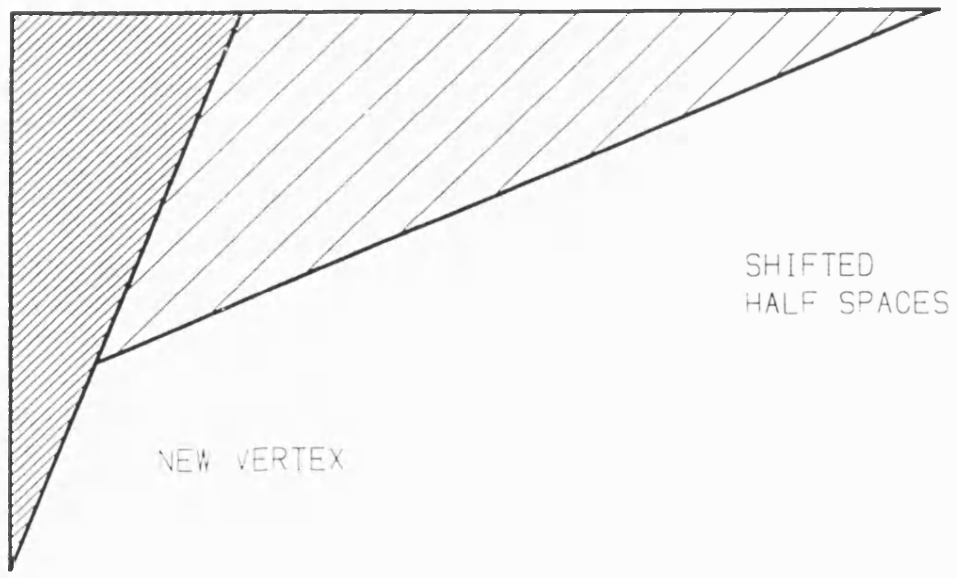


Figure 4.7 Topology changes when redundant intersect-models are grown.

Expression Size	Intersections	Unions	New Size
79	6	33	461
129	7	57	12,623
137	34	34	2,321
187	35	58	54,907
211	41	64	285,397

Table 4.1 Increase in size of set theoretic model expressions after distributing intersections over unions.

Processing such models without further simplification is not feasible, and so methods of simplifying the distributed expression are needed. However, it is difficult to see where such simplification can occur whilst retaining the property of having intersections distributed over unions.

In view of this problem, an alternative was sought which is based on the original set theoretic expression and which detects and corrects any topology changes which occur after shifting half spaces.

4.4 Growing Correction for Facetted Models

4.4.1 Introduction

As mentioned earlier, the topology of facetted models changes when the list of vertices of the model changes. In these models a vertex is an intersection point of at least three planes which has two properties. Firstly, it lies on the surface of the model. Secondly, it is at the end of an edge of this model. When the list of vertices changes, an intersection point has lost or gained one or two of these properties.

An intersection point gains the first property when it changes from lying in air or solid with respect to the model to lying on the surface of the model. This means that the result of the membership test described in Chapter 3 changes and by determining which half space contributions within the test have changed, it is possible to determine which half spaces have changed position relative to the intersection point.

An intersection point will lose the first property when it changes from lying on the surface of the model to lying within air or solid with respect to the model. Again the membership test can be used to ascertain which half spaces have changed position relative to the vertex.

The second property of vertices, that of lying at the end of a model edge, is more difficult to detect and so does not help in detection and correction of topology changes. It can be used to determine whether an intersection point which lies on the surface, *tentative vertex* is a real vertex of the model, or whether it lies on a locally flat region of the model surface.

During the growing process, internal features will contract and, if surfaces are displaced far enough, may disappear. Hence disappearing vertices are to be expected. However, the emergence of new vertices after growing is an indication of unwanted changes in topology, and an algorithm has been developed which attempts to correct topology by eliminating such vertices.

4.4.2 Elimination of New Vertices

Each half space intersection point is uniquely defined by the half spaces which pass through it. Since each half space in the ungrown model has exactly one corresponding half space in the grown model, it is possible to map the intersection points in the ungrown model to those on the grown model. If a list of vertices also exists for both grown and ungrown model, it is possible to construct lists of newly appeared vertices and disappeared vertices.

A half space $P(x,y,z)$ can contribute solid, air or surface to a vertex (v_1, v_2, v_3), and this is determined from the sign of $P(v_1, v_2, v_3)$. A new vertex can only appear because the contribution from one or more of the half spaces in the set theoretic expression has changed. By comparing the contributions of half spaces to the new vertex in the grown model with the contributions to the *corresponding* vertex in the original model, it is possible to determine which half space contributions have changed. Correction of the grown model then involves shifting these half spaces (in the grown model) back towards their original ungrown model locations so that contributions are restored to their original type.

Not all half spaces can be shifted back. The grown model contains surface offset from the original model surface. If the half spaces contributed surface in the original model, then it is likely that they will do so in the grown model, the exceptions being when internal features, such as holes and slots, collapse during growing. Shifting back the "surface" half spaces is clearly incorrect, and such half spaces are classified as *non-shiftable.* Half spaces may, therefore, be categorised as either shiftable or non-shiftable, and corrections are made by only shifting back the shiftable half spaces.

The set of shiftable half spaces is determined from the vertices of the original model. If one of these vertices lies on a half space, then that half space is classified as non-shiftable. The remaining half spaces are classified as shiftable. This test is conservative with respect to identifying half spaces which contribute to the surface since a half space may pass through a vertex without forming part of the model surface. As an example, consider a cylinder unioned onto a cuboid block such that one of the sides of the cuboid passes through the centre of the cylinder, and also such that this cuboid side has the same length as the cylinder diameter [Fig 4.4]. The cuboid side passes through the joining vertices, but does not form part of the surface which is defined by the union of cylinder and block.

As described above, half spaces are only allowed to be shifted one way; that is back along the direction from which they were shifted during growing. The distance shifted depends on the required contribution change. For example, if the contribution required at (v_1, v_2, v_3) is air and the half space $P(x, y, z)$ contributes solid, a λ is required such that the new half space $S_\lambda(P)$ contributes air. Since motion is restricted to moving back along the plane normal, this means finding a $\lambda > 0$ such that $P(v_1, v_2, v_3) + \lambda > 0$, or $\lambda > -P(v_1, v_2, v_3)$. A λ is chosen which is just sufficient to alter the contribution as required.

In some cases, the new vertices which have appeared in the grown model will be at the intersection point of shiftable half spaces and located such that there are no shiftable half spaces contributing to it that may be shifted to correct it. In this case, the vertex itself has to be shifted in order to correct the contributions from other half spaces. The vertex is moved by selecting a shiftable half space on which it lies and shifting it in the same way as described above and relative to the half spaces whose contributions are incorrect.

As changes are made to the model, the list of new vertices is updated. If a vertex lies on one or more half spaces which have been shifted, its new position is calculated. This involves determining the translation which occurs along each plane-plane intersection line passing through the vertex as half spaces are shifted, and then shifting the vertex appropriately [Fig 4.8]. In some cases, this newly positioned intersection point is no longer a real vertex, since shifting a half spaces usually eliminates sets of vertices.

The technique just described works for a large number of examples, but fails for one noticeable case. If two half spaces which both contributed surface in the original model pass through each other, i.e. change relative positions, any shiftable half spaces which used to lie between them will introduce new vertices which cannot be eliminated. One example of this is a groove built from three blocks [Fig 4.9]. The algorithm described above will move the shiftable half spaces back to make the correct contribution, but this is impossible since the position relative to the original surface half spaces no longer exists in the grown model. Moving the half space to correct a vertex will always introduce an new vertex [Fig 4.9].

In summary, eliminating vertices works for many simple examples, such as the crucifix built using differences. Plate 4.1 shows the crucifix and the grown crucifix after correction by eliminating vertices. Plate 4.2 shows the incorrect topology which occurs after growing of the model.

The problems caused by surface half spaces passing through each other seems to suggest a more fundamental algorithm which attempts to maintain the relative positioning of half spaces.

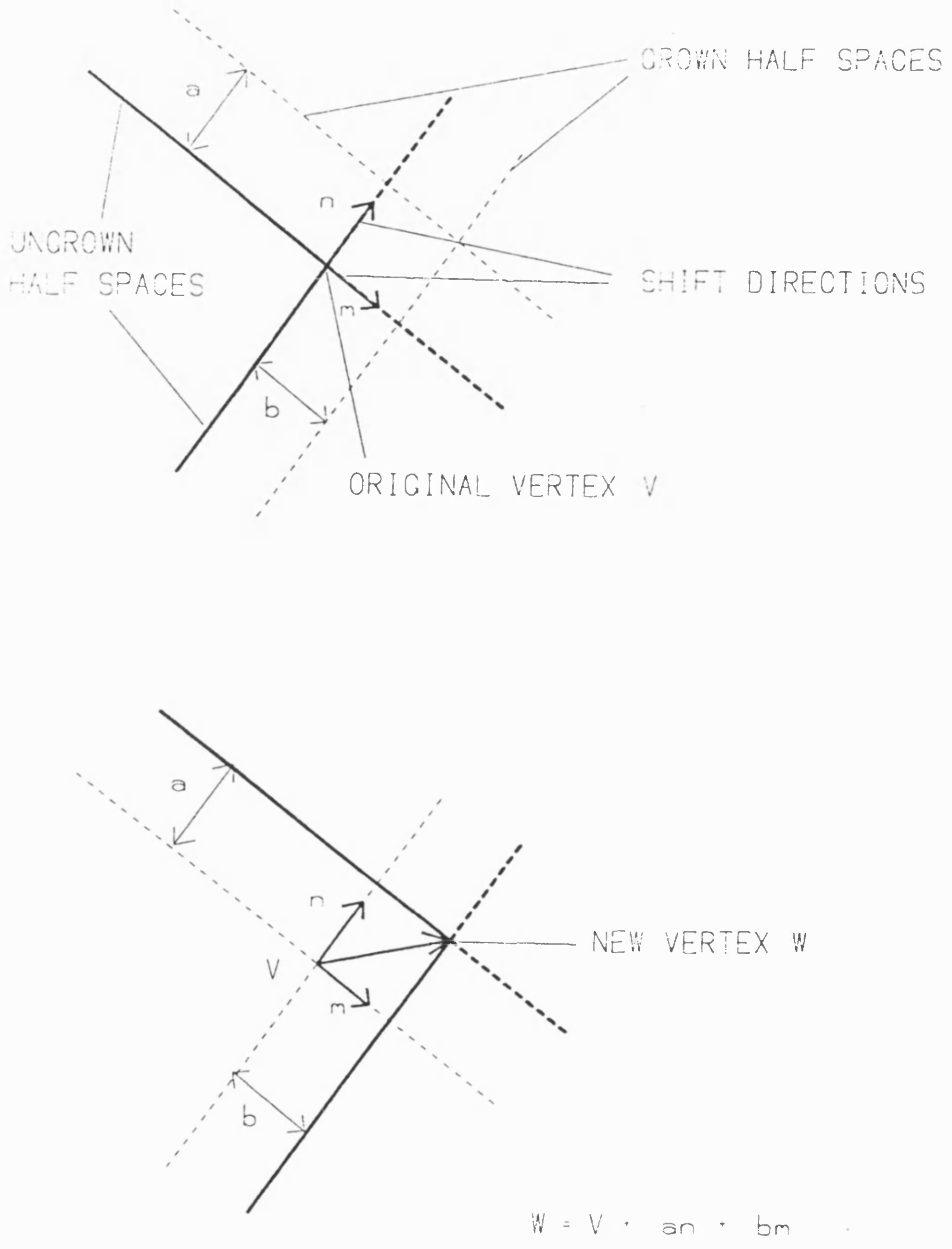


Figure 4.8 Calculation of shifted vertex from original.

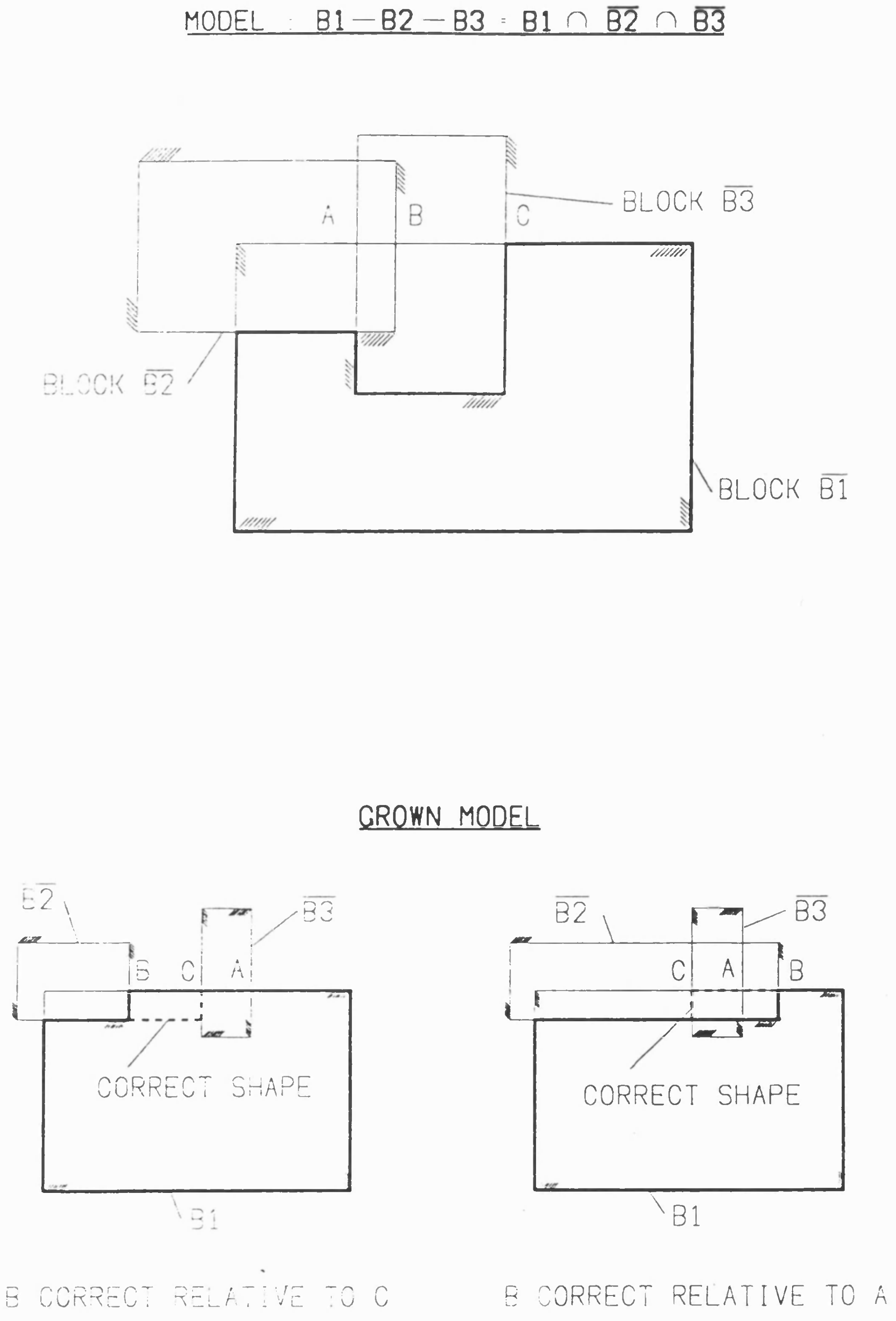


Figure 4.9 Counter example to elimination of vertices algorithm
Since half space C has passed through half space A, the intermediate half space B cannot be corrected by elimination of new vertices. The algorithm will attempt to correct B relative to C (or A) with the results shown.

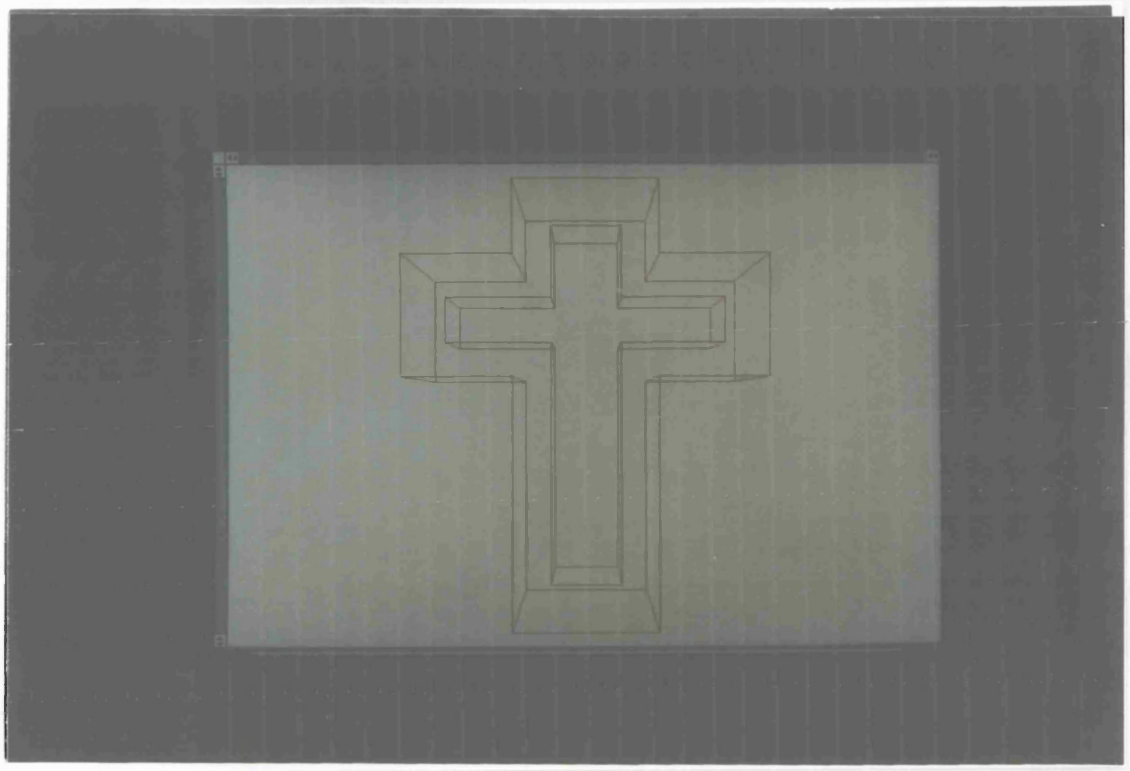


Plate 4.1 Crucifix and grown crucifix. Topology has been corrected using elimination of vertices algorithm.

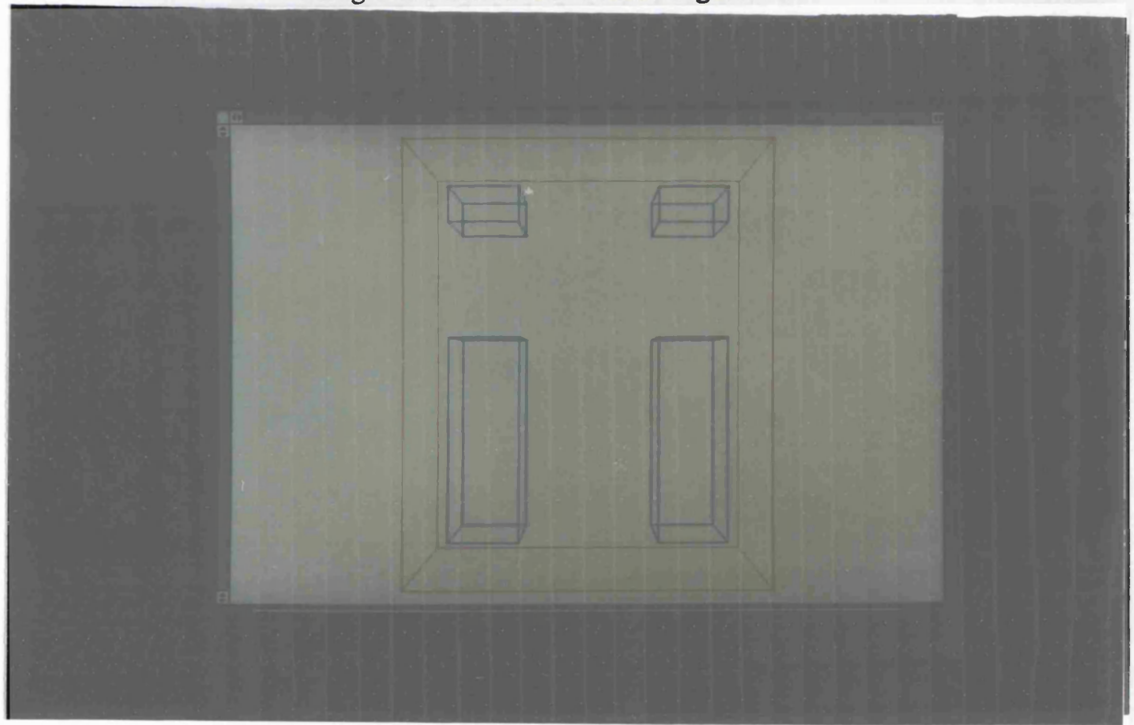


Plate 4.2 Topology changes which occur when the crucifix is grown

4.4.3 Repositioning Planar Half Spaces

The simplest case of half spaces passing through each other is when parallel half spaces are shifted through each other, as in the groove in Fig 4.9, and attention is restricted to this case.

Lists are built, for both grown and ungrown models, of half spaces whose normals lie along the same straight line. These lists are ordered relative to the position of the half spaces along the line, and the differences in relative positioning of half spaces within the two models can is detected. Corrections, made to the grown model, can only be done by shifting shiftable half spaces back along their normals until their position in the list of grown half spaces is the same as their position in the list of the ungrown half spaces.

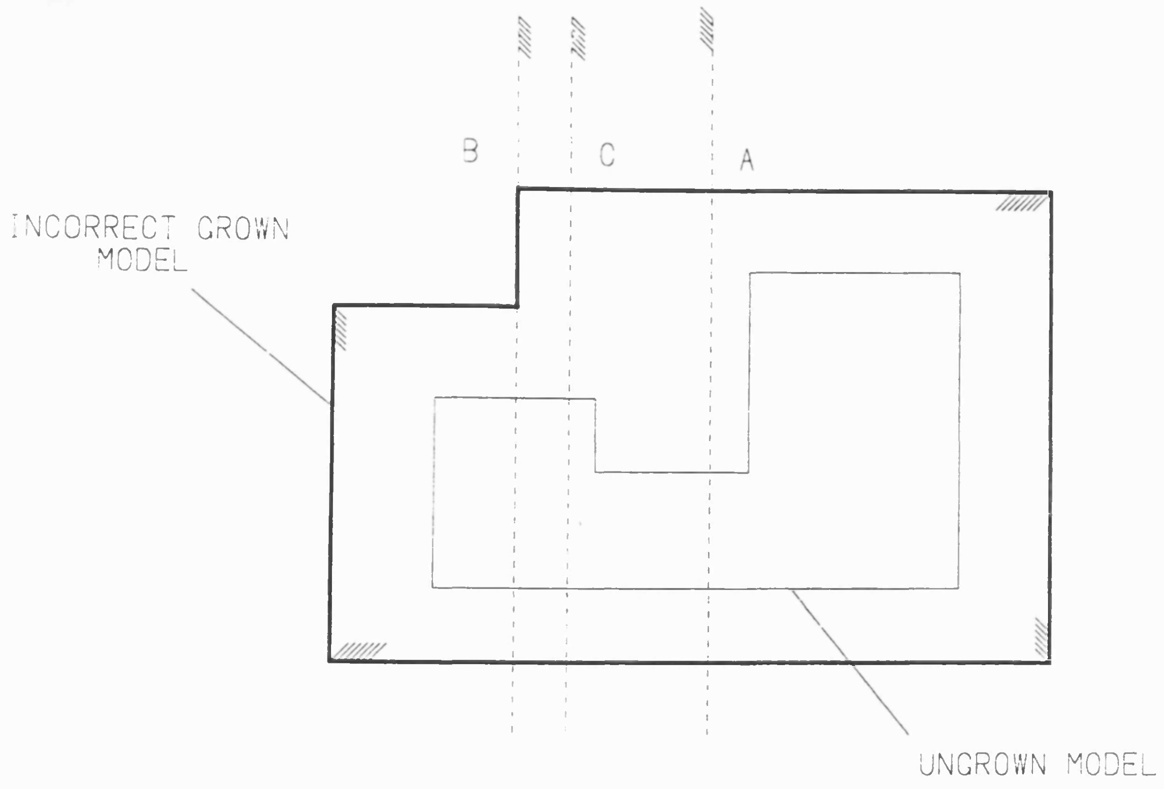
It is easy to detect when two non-shiftable half spaces pass through each other by comparing the corresponding lists. If, in the ungrown model, there are shiftable half spaces lying between such a pair of non-shiftable ones, it is impossible to place them in the same relative positions within the grown model.

To explain why, consider an example. Let an ordered list of parallel half spaces, A, B, C, D exist in the ungrown model, with half spaces A and D being non-shiftable, and half spaces B and C being shiftable. By definition, no part of the non-shiftable half spaces lie on the surface of the ungrown model. Suppose further that on growing the model, half space A passes through half space D. Since the new relative positions of half spaces A and D cannot be altered in the grown model, it is impossible to obtain the original ordering of A, B, C, D by only shifting half spaces B and C.

Topology changes which occur when non-shiftable half spaces pass through each other are unavoidable. They arise mostly when grooves or slots close in on themselves and hence are expected. However, problems can arise when the shiftable half spaces which used to lie between them alter the topology of the model. (This occurs because the intersection points between the shiftable planes and pairs of other planes in the model have changed position relative to the two non-shiftable half spaces which have passed through each other). By repositioning the shiftable half spaces to be at the same position as the non-shiftable half space to which they are parallel, new surface segments can be avoided [Fig 4.10].

Since new vertices can be introduced by half spaces passing through edges, vertices or plane intersection points, this repositioning algorithm, which only considers parallel half spaces, will not correct all possible topology changes. However, the algorithm is effective as a pre-process step to the elimination of vertices algorithm.

Plate 4.3 shows the groove and the result of growing the groove. Plate 4.4 displays the corrected grown model. The crosses mark where new vertices still exist in the correction, and the plate shows the correction due to elimination of vertices and the repositioning of half-spaces. For the former, four new vertices appear and for latter only two occur, as expected when the groove closes.



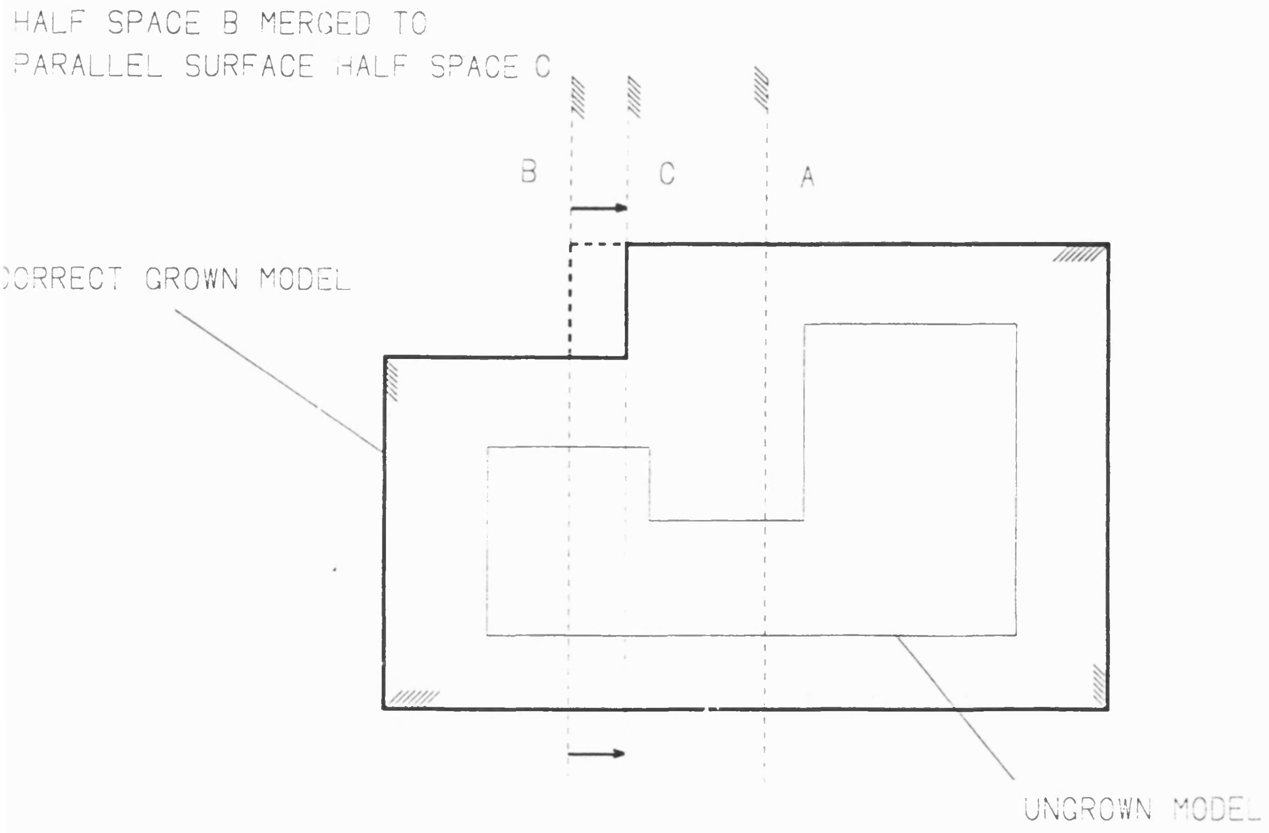


Figure 4.10 Repositioning half-space solution to Fig 4.9
The algorithm merges the intermediate half space to the parallel half space which it use to lie between.

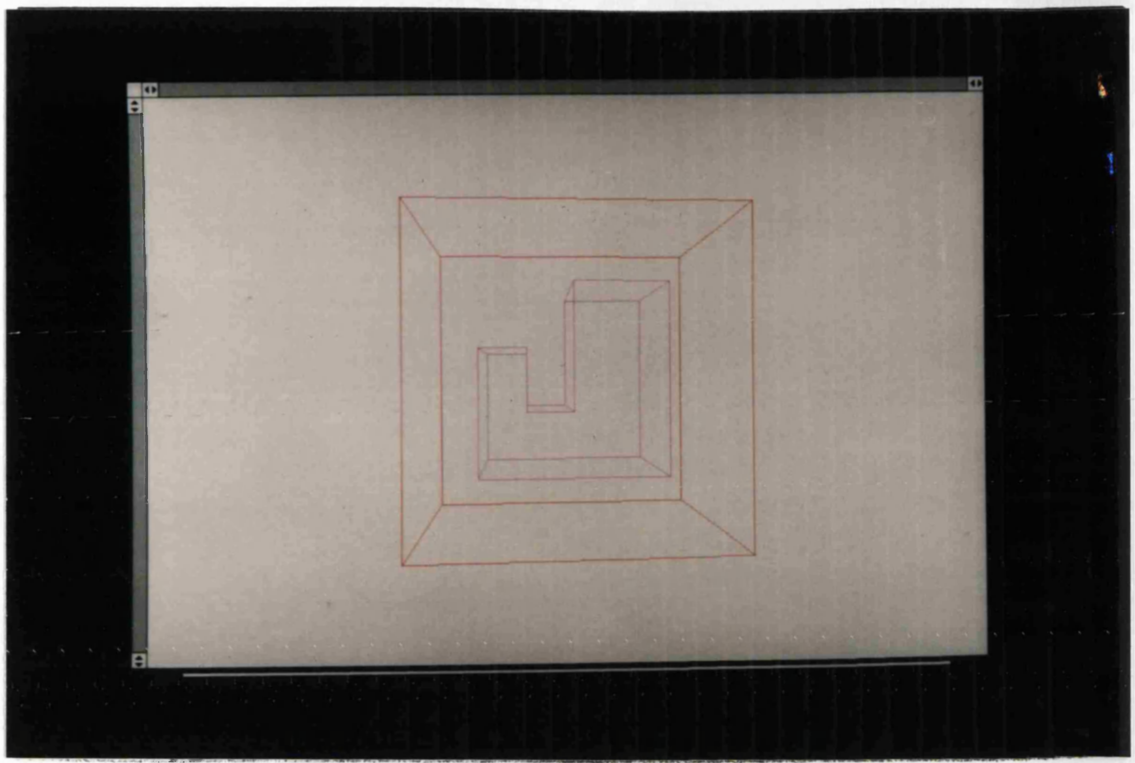


Plate 4.3 Groove and Incorrect grown Groove

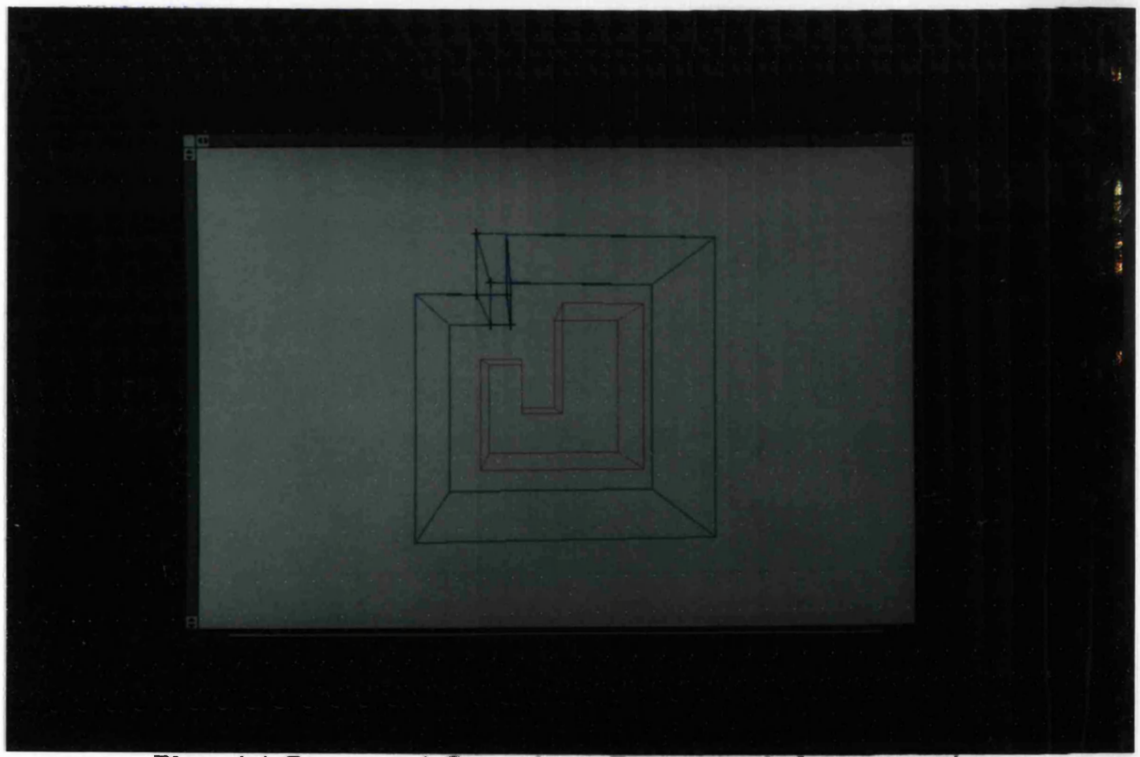


Plate 4.4 Groove and Corrections. Crosses mark the new vertices which exist in both corrections.

4.4.4 An Algorithm for Correction of Grown Models

The correction algorithm implemented uses the repositioning algorithm as a pre-process step :o the elimination of vertices algorithm.

Firstly, difference operators are removed from the model, and the expression is rewritten in terms of intersections and complements.

This model is then grown by shifting the planar half spaces along their normals. A new model is produced with the newly positioned half spaces.

The algorithm considers the grown and ungrown model. Firstly, the non shiftable half spaces are identified by calculating the real vertices of the ungrown model. Secondly, the repositioning algorithm is used to reposition the parallel half spaces within the grown model so that they are in the same relative positions as those in the ungrown model. If any changes are made, a new grown model is created containing these changes.

Next the algorithm generates the vertices of the grown model. Once the vertices are collected, together with their corresponding half spaces, a list of newly appeared vertices is built. The algorithm then attempts to eliminate these vertices by shifting half spaces.

4.4.5 Limitations

So far, no example has been found for which the algorithm has not given correct results. There is potentially a problem of introducing new vertices when half spaces are shifted back during both the repositioning and elimination stages of the algorithm, but this does not appear to happen in practice.

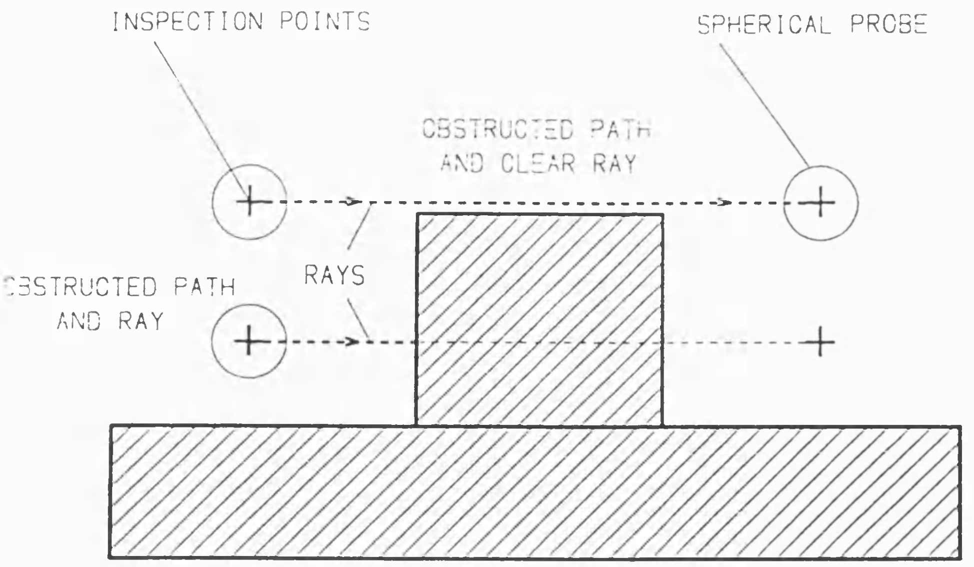
The new vertices which arise due to the union of certain features, like the cylinder and block described above, are not eliminated since both the cylinder planes and the block planes pass through the corners at the union boundary. Hence there are no shiftable planes available to correct the problem.

A majority of topology changes occur when redundant (i.e. not contributing surface to the model) half spaces pass through plane intersection points and thus change their contribution relative to these points. Since these half spaces are often parallel to non-redundant half spaces the restriction of the repositioning algorithm to parallel half spaces is sufficient for a majority of cases. In principle, the repositioning algorithm can be extended to consider half spaces which do not intersect within the object space before or after growing.

4.5 Growing for Measuring Probes

4.5.1 Principle and Methods

The collision detection techniques described in this Chapter 6 are to be used in determining a collision-free probe path for a CMM measuring probe. Ray casting into models along probe path segments determines whether the segment is free from collision for a point travelling along it. This is only useful for detecting obstructions for single point probes [Fig 4.11].



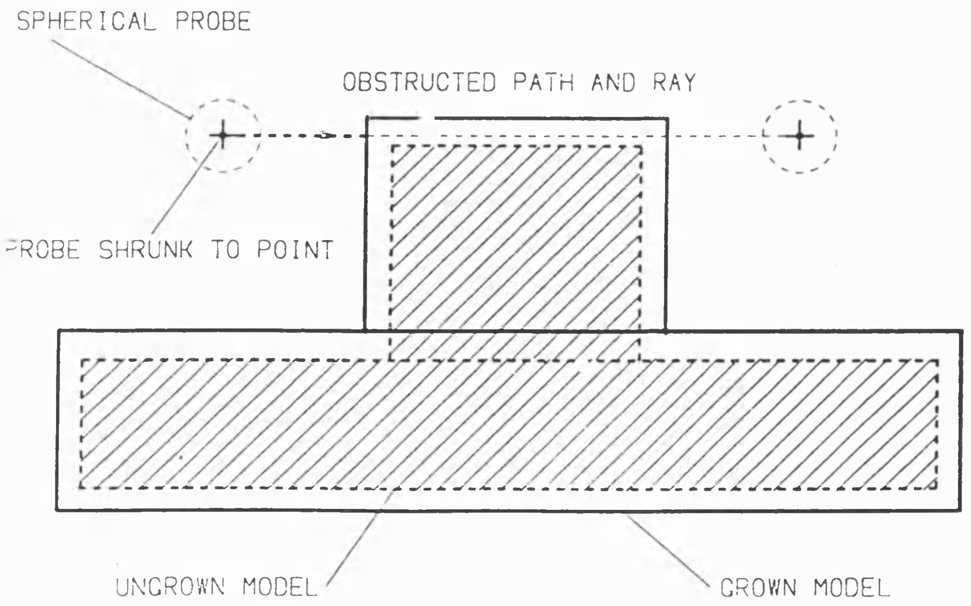


Figure 4.11 Using grown model and ray-casting to detect collisions.

If the measuring probe is modelled as a single sphere, then growing the model by a distance greater than the probe radius enables the probe to be considered as a point with respect to the grown model in collision detection algorithms. Hence, ray casting into the grown model may be used to detect collisions between the spherical probe and the original model [Fig 4.11]. (The distance grown must be greater than the probe radius to avoid collisions which occur because the probe just touches the surface).

The work described in this thesis considers a simple vertical probe, as shown in Fig 5.1. This is modelled as two cylindrical parts and a spherical probe tip. The reverse of the operations needed to shrink a cylinder to a point can be applied to the model to obtain a grown model which can be used in detecting collisions for cylindrical parts of the probe. Two operations are needed to reduce a cylinder to a point. Firstly a shift in the cylindrical axes direction of a distance equal to the height of the cylinder will reduce the cylinder to a planar disc. Secondly, a shift in a plane perpendicular to the axis of a distance which is the radius of the cylinder will reduce the disc to a point [Fig 4.12(a)].

In growing the faceted model for the cylindrical parts of the probe, the two transformations have to be reversed. Thus the half spaces are firstly shifted in a direction along the cylinder axis for a distance equal to the height of the cylinder, and in the opposite sense in which the cylinder was reduced. Next the half spaces are shifted outwards along the components of their planar normals which lie in the plane perpendicular to the cylindrical axis [Fig 4.12(b)]. The order of shifting is unimportant.

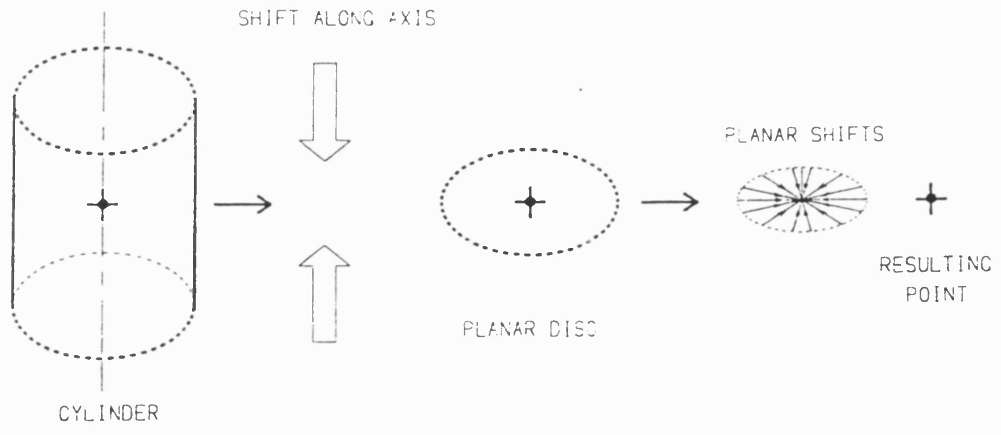


Figure 4.12(a) Shrinking a cylinder to a point.

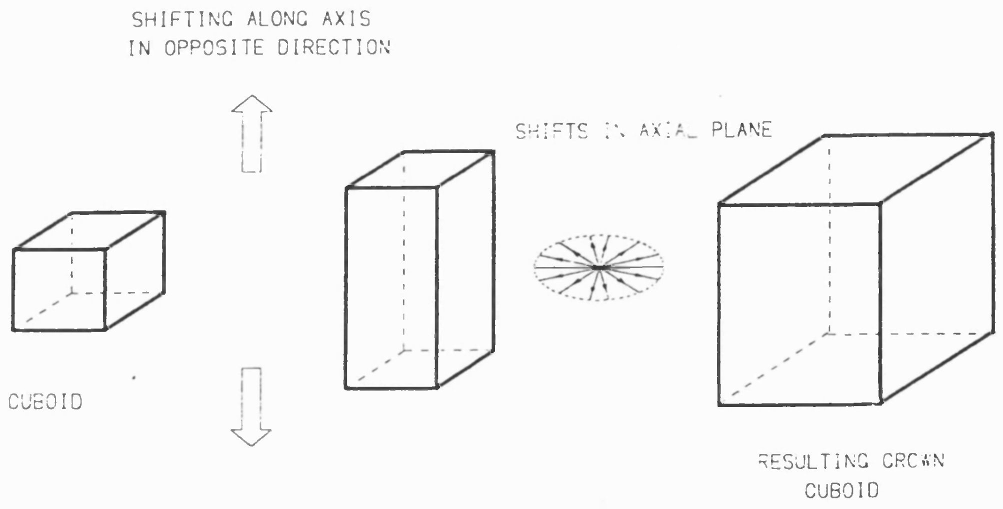


Figure 4.12(b) Growing a block for the cylinder.

For the simple three-segment probe considered in this thesis, each of the separate segments need to be considered in turn and a grown model generated for each part. Hence there will be three grown models, one for each segment. Collision detection techniques will use all three models to ensure that all segments of the probe do not collide with the component.

The growing techniques described for this simple probe can be applied to any segmented probe which has no moving parts (the cranked probe shown in Fig 5.3 for example); by considering each segment in turn and determining the appropriate growing operation. Swivel probes contain a rotatable probing arm which is mounted in a spherical block, the probe head. The probing tip is also spherical. The probe tip and probe head can be considered using the same techniques as for the simple vertical probe described above. For the movable probe arm, a new model is required for each orientation of the arm. This is because the directions of shift needed to reduce the cylinder to a point are dependent on the orientation of the cylinder. The grown model can be calculated by rotating the model so that the arm is a vertical cylinder, and then by applying the vertical cylindrical growing operations.

4.5.2 Examples

The example component used in Chapter 3 is shown in Plate 4.5. The picture shows the result of growing the model for the probe arm by shifting half spaces. The new vertices which appeared are marked as crosses. The cylindrical stepped hole and the groove are destroyed by the growing as previously non-contributing half-spaces are shifted into the model. Plate 4.6 shows the results of running the correction algorithm where the hole and groove are recovered.

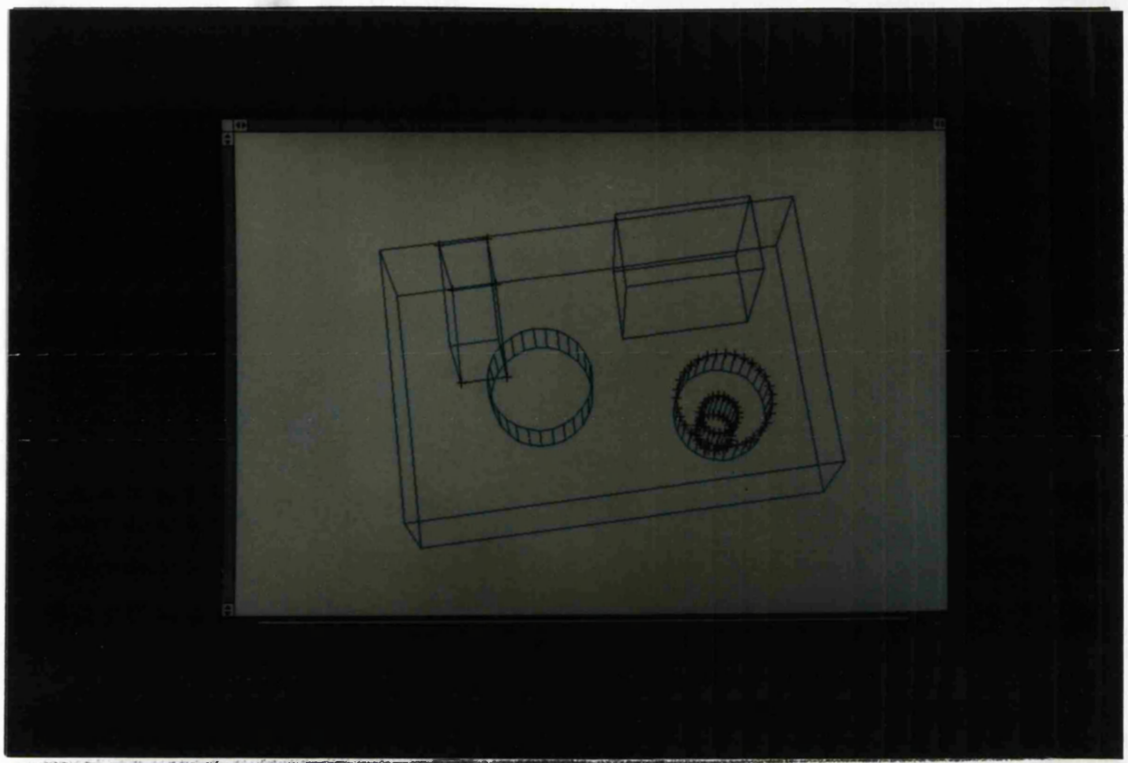


Plate 4.5 The topology changes which occur when a model is grown for the probe arm. Crosses mark where new vertices arise.

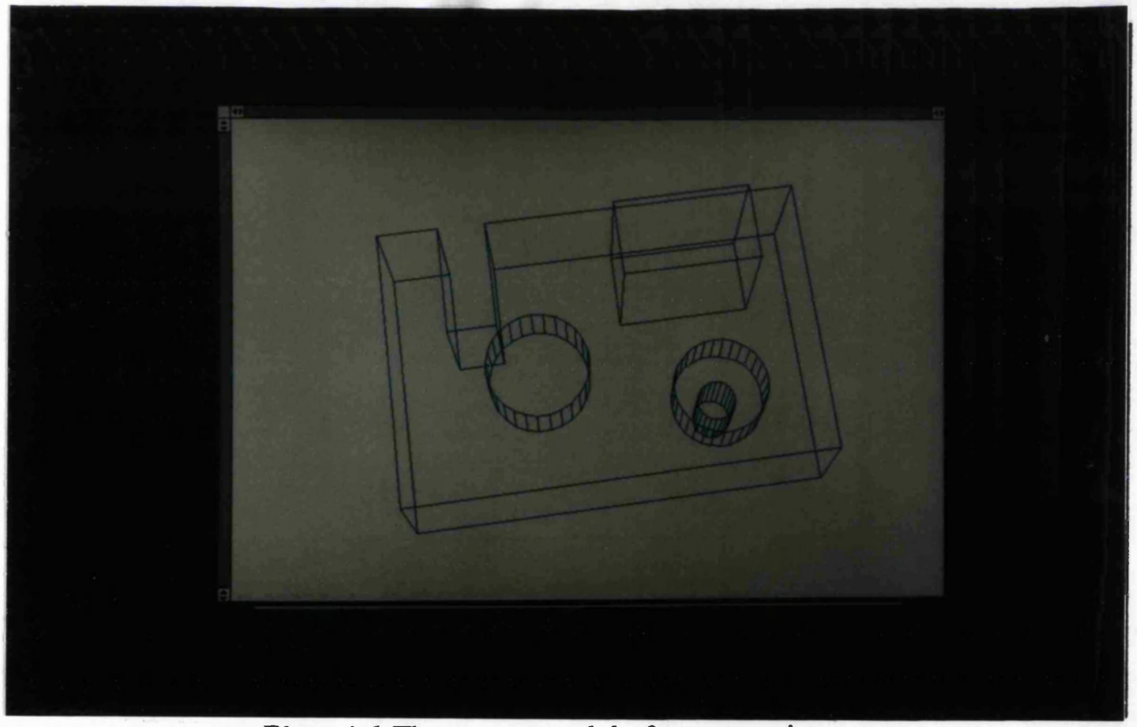


Plate 4.6 The grown model after correction

4.6 Conclusions

Set theoretic reconstruction has been used for various applications, as described in Section 2.6. However the work has dealt mainly with bounded primitives, and uses the properties of bounded boxes to simplify expressions. Such simplification is not available with planar half space primitives.

Growing obstacles for path planning has been used by T. Lozano-Perez and M. Wesley [Section 2.8.2]. Their work restricted attention to polyhedral models and built the grown models by shifting the vertices. Since the objects considered are closed and convex, topology changes are not a problem. Growing set-theoretic models in effect extends this idea to include a broader range of shapes and, although consistent shapes are guaranteed, extra care has to be taken in maintaining the expected topology.

The growing algorithm in this chapter involves shifting planar half spaces, and this restricts its applicability to facetted models. An alternative approach is required for curved surfaces. However, it is possible to calculate the equations of grown half spaces for simple quadrics (i.e. cones, cylinders and spheres) given the ungrown half space equation, and this will enable the majority of engineering applications to be considered using a similar algorithm. Restricting to facetted models, however, also makes correction to changed topology possible and the algorithms implemented in this chapter have been successful on all trials.

In conclusion, it is worth noting that topology changes usually occur when a model has been defined untidily and contains an excessive amount of half spaces which do not contribute surface. For example, if a crucifix is modelled as the union of two blocks, growing will not lead to topology changes. However, if it is defined in the inferior way of differencing four blocks from each corner of a fifth, then extra half spaces which are redundant in the model will create problems when the model is grown. This suggests that efficient modelling approaches will lead to less problems when growing models.

CHAPTER 5

SELECTING THE INSPECTION POINTS

5.1 Introduction

In manual CMM programming, the selection of points for inspection is done by the technicians who program the CMM. They base their selection on experience and on the type of features and tolerances being checked. There are no rigid rules, since the selection depends on the purpose of the inspection task. For the work described in this thesis, the inspection results are to be used in the checking of geometric tolerances.

The techniques described in this chapter process the surface points generated using the algorithms described in Chapter 3. Two types of processing are done.

Firstly, a technique is described for selecting a set of points which can be reached by the CMM measuring probe. Whether a point can be reached depends the location of the point, on the geometry of the probe and on the path of motion used to reach it. A simple vertical probe is considered, and this is modelled as two vertical cylinders and a sphere. Section 5.2.3 suggests how the techniques developed can be extended for more complex probes. The actual path of the probe is described fully in Chapter 6 and Appendix 1. In summary, the probe will be driven to a point which is offset from the surface. The offset distance is specified as a parameter to the system and the offset direction is the negated probing direction which is derived from the surface normal at the point being measured.

The desired properties of inspection point sets were described in Section 3.2. The second type of processing described in this chapter is the techniques used to select a point set with the required properties from the set of reachable points.

5.2 Detecting and Eliminating Inaccessible Points

5.2.1 An Algorithm for Detection of Unreachable Points

The simple measuring probe is modelled as two vertical cylinders and a sphere. A wide cylinder represents the probe head which supports the probing arm, which is represented as a thin and long cylinder, and the probing tip, which is represented by a small sphere [Fig 5.1]. The dimensions are chosen to enclose the real probe, which contains a conical probe arm and a stepped cylindrical probe head.

Not all surface points can be reached by the probe. For example, points lying on faces whose normal is opposite to the positive Z-axis, such as undercuts, are clearly impossible to reach using the probe described above. Further, some points may be obscured by features which obstruct any one of the probe segments. They may be too close to another surface, or too far down a narrow hole or slot for the probe to reach them [Fig 5.2]. An algorithm which detects such points is described which uses the techniques of growing set theoretic solid models introduced in the previous chapter.

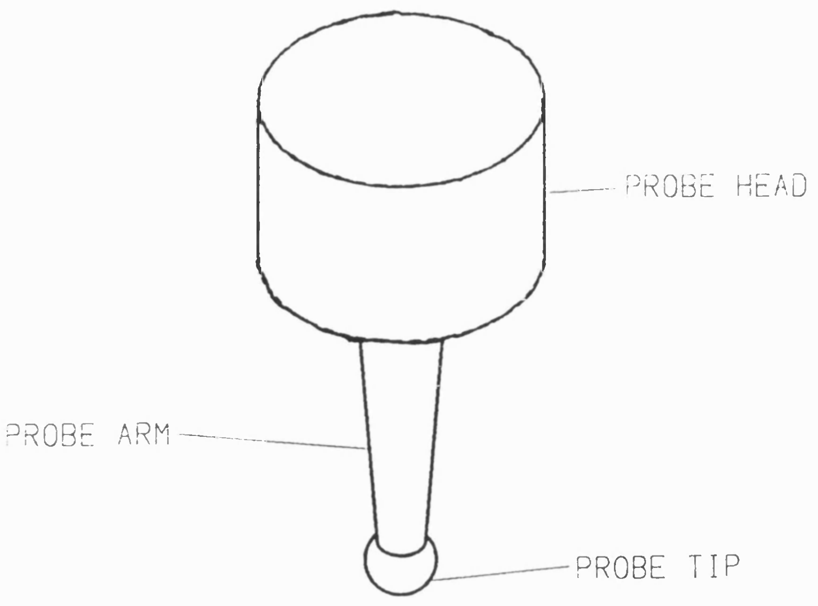


Figure 5.1(a) Vertical CMM Measuring Probe.

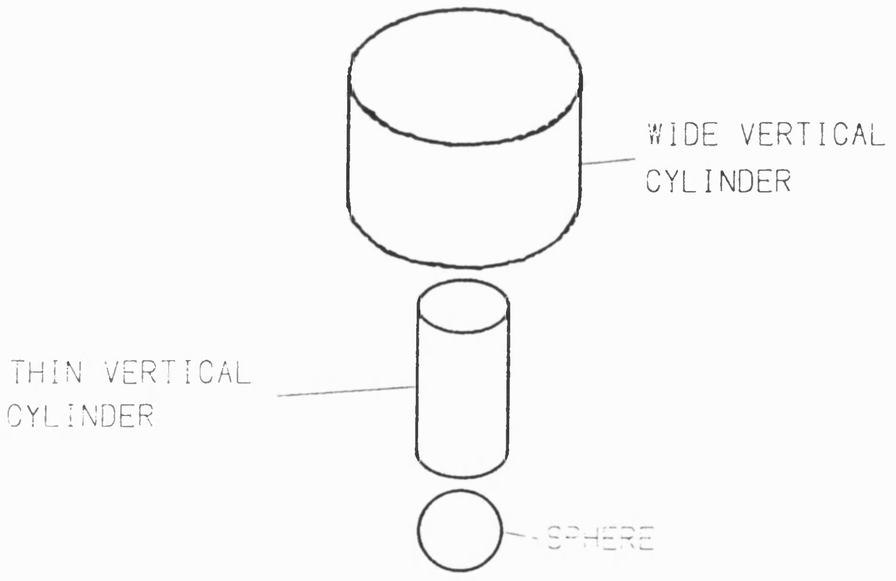


Figure 5.1(b) Exploded model of the probe as two cylinders and one sphere.

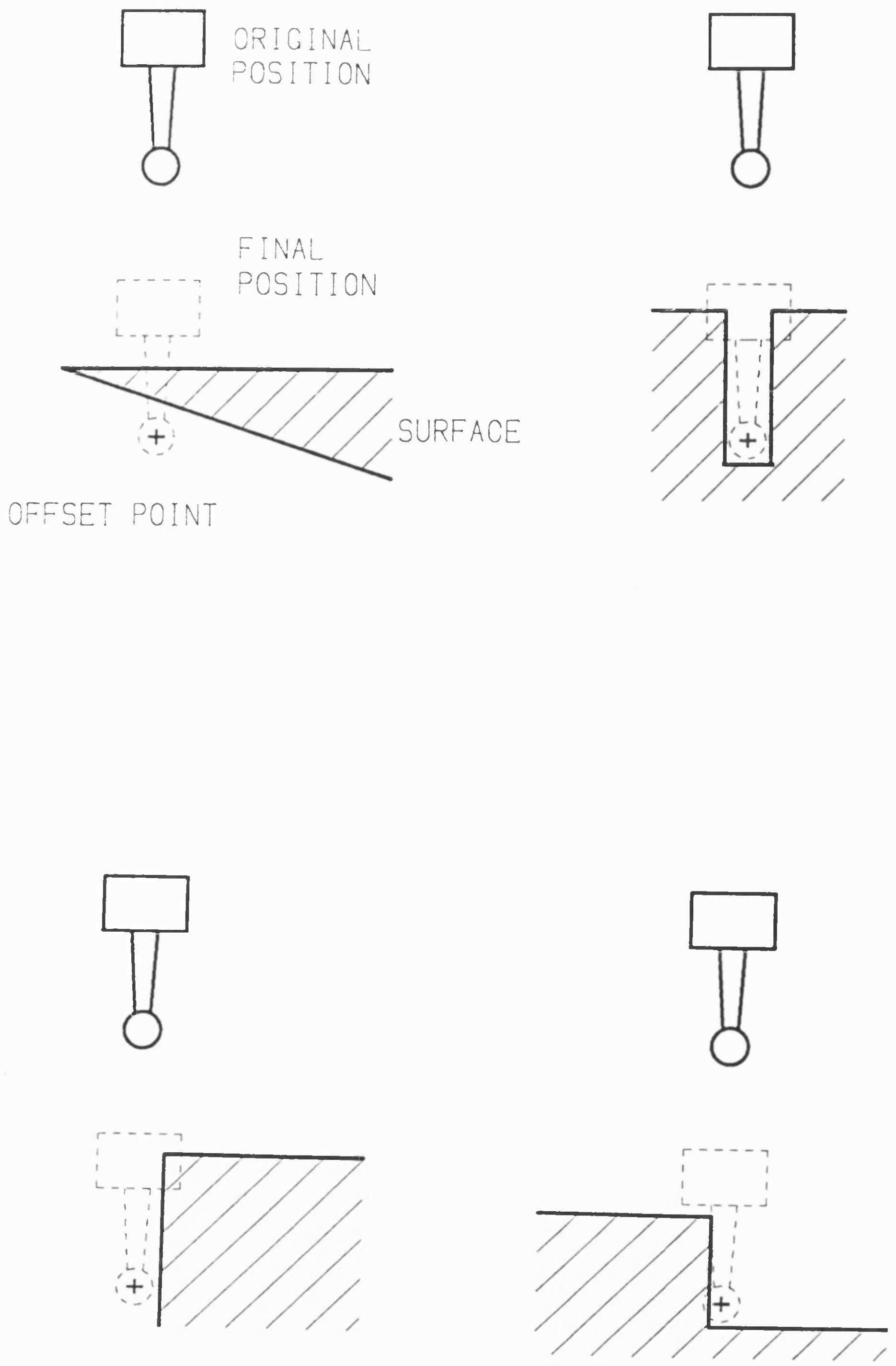


Figure 5.2 Points which are not accessible to the probe.

A grown model is created for each of the probe segments; thus there are three models, one for the sphere, one for the thin cylinder and one for the wide cylinder. The growing transformation results in shrinking each probe segment to its centroid point. Hence the sphere is shrunk to its centre, and the cylinders are shrunk to points lying at the midpoints of their axes.

During measurement of a component, the probe is driven first to a point in air which is offset from each surface point and then driven towards the surface point. This two step motion requires two tests to be made for each surface point. Firstly, the offset point should be checked for probe accessibility. Secondly a test is required to make sure that in moving from the offset point to the surface the probe does not collide with any other surface. This is only a real problem, however, when the direction from which the probe approaches the surface, i.e. the probing direction, is not vertical and so only these cases need both tests.

It follows that either one or two tests may be required for each surface point. To test accessibility of the offset points for the vertical probe, a test is made to determine whether the probe can move vertically downwards to the point from a z coordinate plane which is known to be above the component. This *safe plane* is defined as an attribute of the component model (see Chapter 7). To detect a collision in moving from the safe plane to the offset point, a ray is cast downward from the safe plane (into the grown model) to the offset point. If no surface is hit, the second test is carried out if required. This involves casting a ray from the offset point along the probing direction and checking that the hit surface is the one on which the original surface point lies.

The ray casting is done into each of the grown models. However, the tests are for the position of the centroid of each probe segment. For the sphere the tests can be made directly to the offset points since the probe tip will pass through these points during inspection. For the cylinders, the tests need to be carried out for the points through which the cylinder centroids will pass as the probe moves. These points are calculated from the original offset point by the appropriate translations which, for the simple probe, are parallel to the z axis. A parameter file is used to hold the various dimensions of the probe and to consequently derive the appropriate length of translation.

5.2.2 Results of Applying the Algorithm to Examples

Plate 5.1 shows results applied to the model consisting of blocks and vertical cylinders built on and into a rectangular block which was shown in Chapter 3. The points have been generated using different grid units for different features and only the protruding block and stepped hole are considered. Points inaccessible to the probe in moving from the safe plane have been removed. These included those points too far down the protruding block and cylindrical hole, as well as those points lying on the stepped hole ledge too close to the sides of the hole. Where access was denied by the protruding block to the wide cylinder, such as the side of the stepped hole closest to the block, points were also removed.

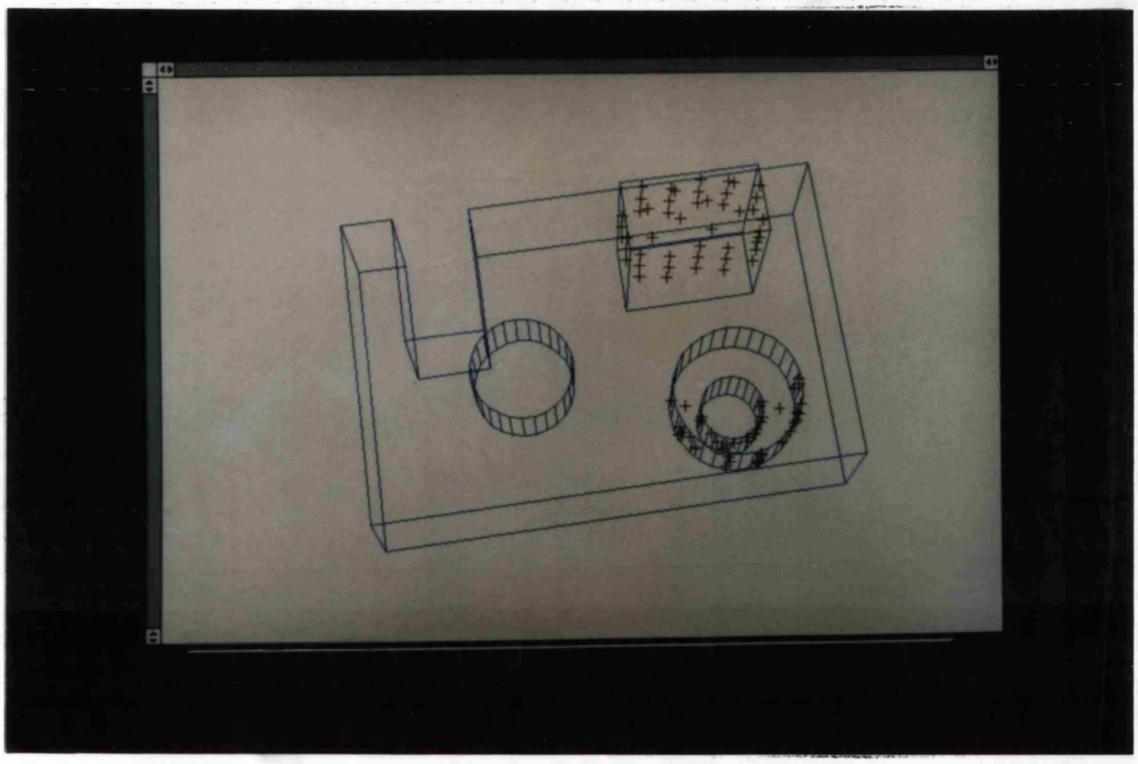


Plate 5.1 Reachable Points for test piece protruding block and stepped cylindrical block

5.2.3 Extending the Algorithm for General Probes

The probe modelled above is one example of many kinds of probes in use. Here we discuss how the algorithm could be extended to three commonly used alternative probes.

A probe whose probing arm is cranked, so that undercuts can be measured [Fig 5.3(a)] can be modelled as four segments, with two cylinders used to model the cranked probing arm, one vertical and one horizontal [Fig 5.3(b)]. The tests for collision detection for the two vertical cylinders are similar to those for the simple probe. Since the surface inspection points define the position of the probing tip, the transformation required to obtain the corresponding position for the centroid of these vertical cylinders must now take into consideration the length of the horizontal arm of the probe.

However, applying the same method to the horizontal cylinder would result in the elimination of points which are reachable, such as those on an undercut. A collision would be detected as the probe arm moved vertically downwards, but in practice the point would not be reached by such a vertical displacement. Instead, the probe would be driven to the point lying on the undercut in two or more steps. For example, by moving the arm vertically downwards along a known clear path and then moving the probe towards the point in a plane perpendicular to the Z-axis [Fig 5.4(a)].

By testing for the vertical cylinders, the points on the undercuts which genuinely cannot be reached are eliminated. This is because if collision between the probe head or the vertical arm occurs when the probe tip is positioned at the required point, then the point cannot be reached [Fig 5.4(a)].

In order to prevent elimination of reachable points, special procedures can be used whenever such an undercut occurs. These would start by carrying out the test for the spherical probe tip as described in the previous Section. If no collisions occurred, then the test for the horizontal probe arm could be done against the appropriately grown model as with the simple probe. This test would separate points which lie on an undercut from those which do not.

If a collision for the probe tip is detected, a ray could be cast from the point of collision on the surface along a direction parallel to the axis of the horizontal cylindrical probe arm and towards the vertical probe axis. By collecting the intersection points of this ray and choosing the last point at which the ray leaves solid before hitting the axis, a clear point for the probe tip could be found. This point could then be used to determine a new test offset point for the horizontal cylinder [Fig 5.4(b)]. The vertical cylinders would also need to be checked again at this new point. If no such point exists, or if the subsequent checking detects collisions, then the original offset point is unreachable.

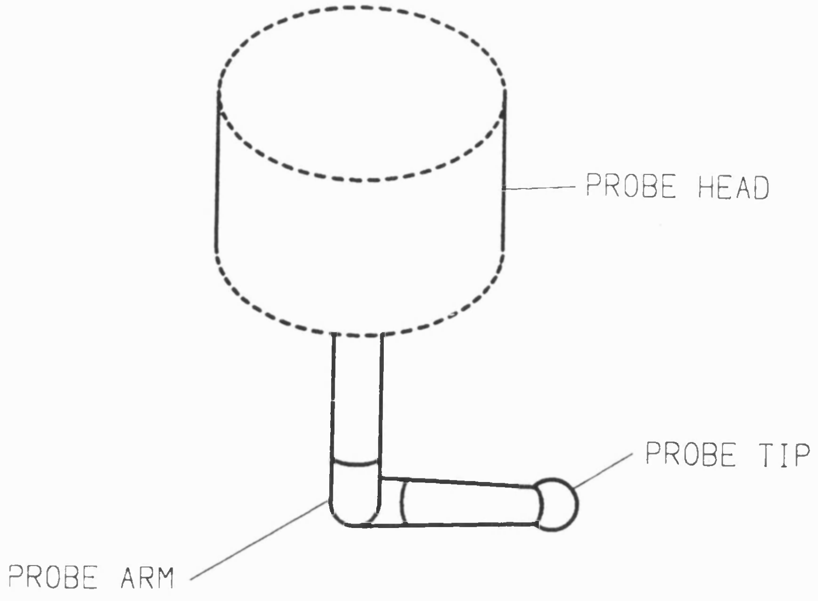


Figure 5.3(a) Cranked Probe.

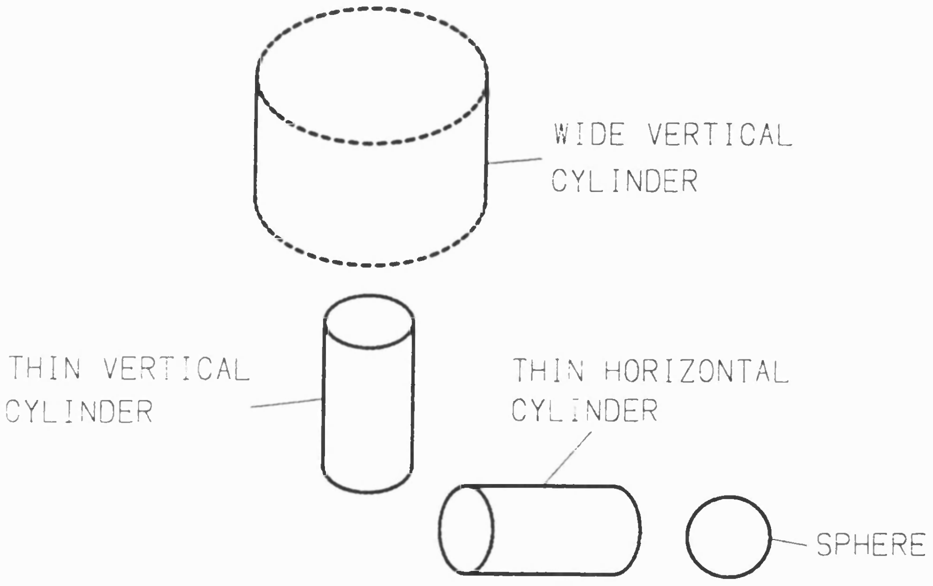


Figure 5.3(b) Exploded model of the probe as three cylinders and one sphere.

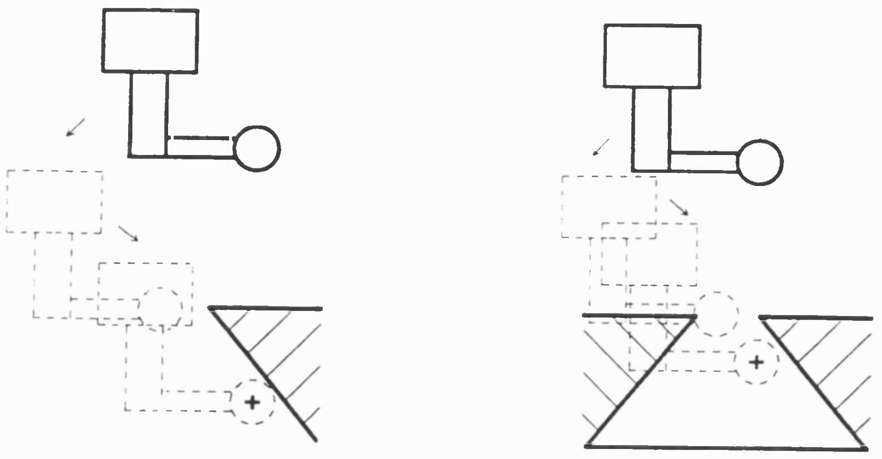


Figure 5.4(a) Reachable and unreachable points lying on undercuts.

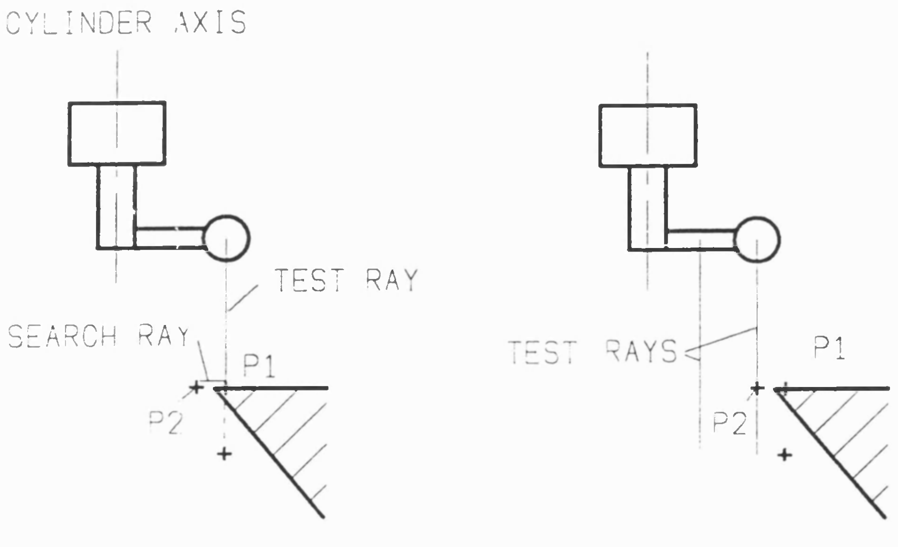


Figure 5.4(b) Detecting reachable points on undercuts using test rays.

The above procedure effectively attempts to determine the existence of a path segment from the safe plane to the offset point. For a majority of the points, the tests described for the simple probe are sufficient for detecting unreachable points, and only surfaces whose normal faces downwards will require such procedures. Further, since the decision about whether a probe can reach the point or not depends on the geometry of the probe, the procedures will necessarily be probe specific; it is difficult to derive general procedures which can handle all types of probes.

The multi-probe tip, which consists of several horizontal probes and a vertical probe, is another commonly used probe. An algorithm which uses the procedures described above and those in the previous section can be developed for this case, assuming that a specific probe has been selected for the point being checked. Such a selection could be made based on the probe geometry and the surface normal at the point. An alternative would be to test all probes and select one, if any, which can reach the offset point.

Finally, another common type of probe in use is the swivel probe, which consists of a spherical probe head supporting a cylindrical probe arm and spherical probe tip. The arm can be orientated at various angles. The accessibility for the probe head can be checked in the usual way for each point. The offset points calculated for the centroid of the spherical head will be dependent on the orientation of the arm. For the swivel probes currently in use, the number of allowed orientations is finite and by using the grown models, a different one for each orientation, and testing the three swivel probe centroid points, a collision-free position for the probe can, in theory, be found. Information about the surface normal will help in reducing the number of orientations to try.

Once it has been established that the probe head can be positioned correctly, the next problem is to determine whether there is a collision-free path for the swivel probe from the safe plane. The solution to this is non-unique and an algorithm is required which determines a path segment consisting of translations and rotations. Algorithms exist for path-planning of robots which involve solving problems such as these (see Section 2.8). The grown model at each orientation will help in the collision-detection stages of these algorithms.

5.3 Selection of Inspection Points

5.3.1 Distribution Considerations

As described in Section 3.2, the distribution of inspection points is an important factor in the control of the success of inspection. To summarise, the distribution should not contain groups or clusters of points but should be sufficient to be representative of the surface.

The generation of surface points described in Chapter 3 uses grid techniques together with perturbation parameters to obtain a distribution with the required properties. Elimination of points which cannot be reached, however, may remove some of these properties, and result in the undesired localisation and clustering of points described above. A post-processing algorithm is used to de-cluster these points in an attempt to improve the distribution.

The algorithm builds a list of point clusters for each feature. Each clusters is built by calculating the centroid of the cluster and only adding new points if they are within a given distance of this centroid. The centroid is updated each time, and if d is the distance used, all points will approximately be within $2d$ of each other.

Given the list of clusters, surplus points can be eliminated by only selecting one inspection point from each cluster. This will generate a de-clustered point set but care is needed to avoid an unacceptable reduction in the number of points for a feature.

A problem with eliminating points in this way is that the resulting point set may no longer be representative of the surface. For example, if the point set consisted of four points for a plane inspection task and three are co-linear, elimination of the fourth point would render the set of inspection points useless for planar analysis. This example extends directly for quadrics where elimination of the wrong points will results in co-planar point sets. It is trivial to check for these cases before points are eliminated.

In general, however, when given a choice of points to eliminate, the aim should be to obtain a set of points which enables most benefit from subsequent analysis. This depends on the kind of analysis which is to be carried out and the algorithms which will be used. For example, if least-squares fit algorithms are used in the analysis of measurements, then the effect each point, or subset of points, has on the accuracy of these surface fitting algorithms can be used to select the best set of points. Such selection will rely on statistical methods based on detection of outliers or influence points.

5.3.2 Quantity Considerations

The number of points is clearly an important factor in inspection; too few and the data will not be representative; two many and the inspection becomes inefficient.

The number of points depends on the inspection task and the feature being inspected. If a tolerance of form, such as roundness or flatness, is being checked where the shape of the surface is important, then the more points the better, subject to efficiency of inspection.

If a tolerance of attitude or location is to be checked, then fewer points are needed subject to the minimum for the particular feature. For example, three points are necessary for a plane. The BS7172 standard offers guidelines for the required number of points in assessing geometrical errors (see Section 2.7).

The declustering algorithm uses the number of points suggested in BS7172 to determine an optimum size of the cluster. The algorithm starts by clustering points subject to:

$$d = \text{(point grid spacing for feature/2)}$$

If the number of points in the point set is more than the required number, N, but the number of clusters is less, an iterative procedure is used to determine the d which gives the closest number of clusters to N. The procedure reduces d by half at each iteration until the required number of clusters is obtained or the distance is less than a specified limit.

Similarly, if the number of points and clusters are much greater than the suggested number an iterative procedure is used which increments d, by an increment i, until the required number is obtained. (In some cases both the parameter d described above and the parameter i may be used to obtain the required clustering, with the algorithm terminating when either or both parameters become smaller than some specified limit).

Once the desired number of clusters have been found, a point is chosen from each cluster to yield a distribution with the required properties.

5.3.3 Examples of Declustering Points

Plate 5.2 shows the results of declustering the points shown in plate 5.1. The clusters have been removed and since only three of the five faces of the protruding block have been toleranced, only points on these faces remain.

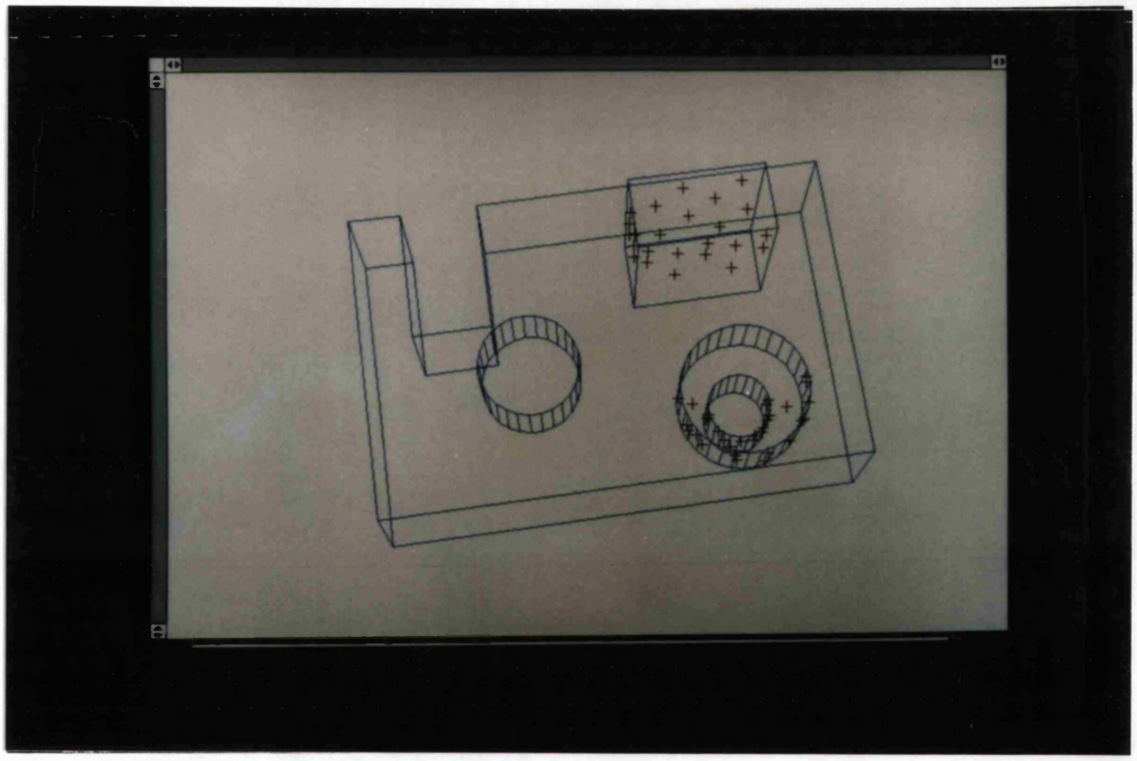


Plate 5.2 Declustered set of points for object shown in plate 5.1.

5.4 Conclusions

The techniques of collision detection using grown models can be used to determine whether or not points can be reached by a measuring probe. The complexity of such a problem, however, depends on the geometry of the measuring probe and its number of degrees of freedom. The algorithm described has been implemented for the simple three-segment probe and can be extended for many other types, although it is less efficient for use with the swivel probe.

The problem with the swivel probe is the need to generate a collision-free path from one point to another for a probe which can pass through a number of fixed orientations. This is analogous to the problem of robot path-planning which moves a robot from one orientation and position to another. The problem can be simplified if it is assumed that the probe only changes orientation above the z- safe plane with motion to the offset points only consisting of translation, as with the simpler fixed probes. The techniques described above then can be used for each fixed orientation of the swivel probe.

Little work has been done on automating the selection of inspection points. British Standard BS7172 offers guidelines for the selection of points for simple features such as planes, spheres, cones and cylinders and these guidelines have been followed in choosing the number of points in the algorithms described.

The distribution and number of points can effectively be controlled after point generation by the elimination of points. Since elimination tries to obtain the suggested number of points for each feature, the original quantity is unimportant and hence, in general, it would be beneficial to start with an excessive number and allow the post-point generation algorithms select an appropriate subset.

The problem of finding the best distribution of points for the inspection of features is non-trivial. The influence of each point or each set of points on the whole distribution needs to be assessed, with influence being measured in terms of the effects the points have on the subsequent analysis. The points which have least influence can be omitted. However, eliminating a set of points may have a different effect on the distribution than eliminating each point individually, and so the problem becomes more complicated. Further, it is not clear how to measure the influence of point sets for, say, simple analyses such as least squares fit, and thus such techniques have not been developed in the project described here.

CHAPTER 6

GENERATING THE MEASURING PATH

6.1 Introduction

Chapter 3 and Chapter 5 described algorithms which take as input a solid model of a component and generate a set of surface points for use in CMM inspection of the component. In order to measure the component at this set of surface points, a CMM measuring probe is driven around the component. Full details of collecting CMM measurements are given in Appendix 1. In order for the CMM to reach the inspection points, a three-dimensional path which passes through the coordinates of points offset from the surface points is required. This measuring path should be time-efficient and also must avoid unwanted collisions with the component. This Chapter describes an algorithm for detection of such collisions and illustrates how such an algorithms is used to construct an inspection path for the probe.

Currently, such paths are decided by skilled technicians. They are either programmed off-line as a part program or manually taught to the CMM. Both methods are tedious and time-consuming, and also prone to time wastage through human errors. This makes the path generation stage of inspection an obvious candidate for automation.

This chapter describes an algorithm for automatic path generation which use set theoretic solid model representations of components. Two major problems are addressed. Firstly, a method of finding a time-efficient path through the surface offset points is discussed. Secondly, a technique is described for detecting unwanted collisions and modifying the probe path to avoid them. A simple vertical probe is considered and collisions are detected using the model growing and ray-casting methods described in Chapter 4, with tests being made for each segment of the probe. This does restrict attention to faceted models, but given that curved surfaces can be approximated to any degree by facets, this does not limit the techniques described.

6.2 Determining Time-efficient Paths

6.2.1 Cost Function

In order to assess the time-efficiency of a path, the time taken in moving the probe between points needs to be estimated. The cost function is a calculation which does this for two points by calculating the cost of moving between the two points. Such a function can be complicated, including factors specific to the CMM and/or its environment. In this thesis a simple model has been used which only considers the distance to be travelled between points; more complicated models can be derived as part of future research.

The time taken for the CMM probe to move between two points is a function of the distance travelled by the probe; the greater the distance, the longer the time. Thus, if point P has coordinates (x_1,y_1,z_1) and point Q has coordinates (x_2,y_2,z_2) then an idea of the time taken, or *cost*, in moving from P to Q is given by the distance between the two points, d where:

$$d = \left((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 \right)^{\frac{1}{2}}$$

If a collision is detected when planning a direct path from P to Q, then extra path steps will be needed to move around the obstruction. This will affect the cost value, which should be modified to include the extra distance travelled. An algorithm for moving around obstructions is discussed in Section 6.3.2. Such an algorithm can be applied at this stage to determine the cost.

A conservative estimate of this cost can be obtained by using the z-safe plane which is known to be above the component described in Chapter 5. The cost of moving around the obstruction can be estimated by calculating the total distance travelled in moving from the first offset point to the second offset point via the safe-plane. This is a three step motion. In step 1 the probe moves vertically upwards to the safe-plane. In step 2 the probe moves in the safe-plane to a point above the destination point. In step 3 the probe moves vertically down to the destination point [Fig 6.1(a)]. Both the offset points are known to be reachable from the safe-plane because of the method of their selection described in Chapter 5. The estimate will be conservative since it employs an algorithm which is independent of the component geometry [Fig 6.1(b)].

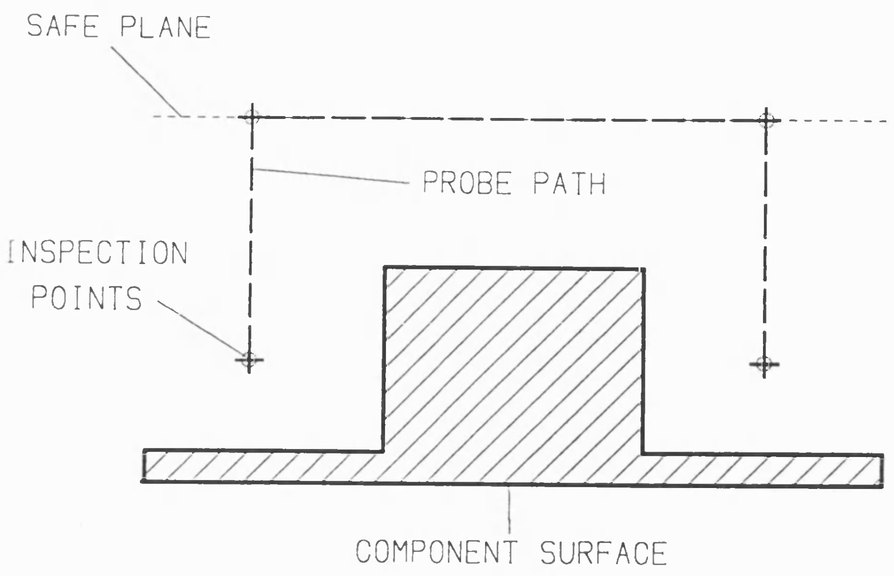


Figure 6.1(a) Collision avoidance using safe plane.

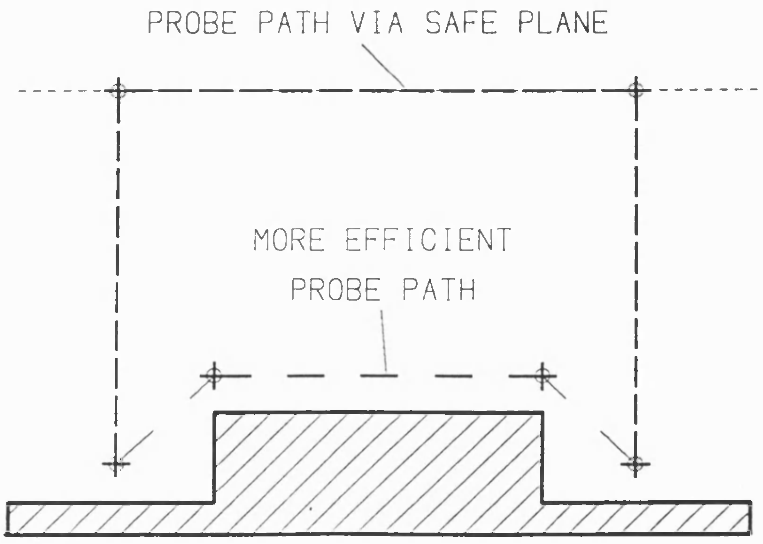


Figure 6.1(b) Inefficiency of safe plane steps.

If, for the CMM, a particular coordinate axis direction has a greater acceleration or velocity than another, then motion in this direction should be encouraged to reduce the time taken for motion. The cost function can be modified to reflect such preference by using coordinate weights, w_x, w_y, $w_z > 0$ to bias the cost for a particular axial direction. The cost estimate, d, then becomes:

$$d = (w_x(x_2 - x_1)^2 + w_y(y_2 - y_1)^2 + w_z(z_2 - z_1)^2)^{\frac{1}{2}}$$

6.2.2 The Travelling Salesman Problem and Path Generation

The measuring probe must pass through every offset point, and as shown in the previous section, motion between each pair of points has a cost associated with it. An efficient path would minimise the total cost of the journey, calculated by summation of the cost of each path segment. This is the well known travelling-salesman problem, where a salesman must visit all cities such that the total distance travelled is a minimum.

After consideration of various algorithms [see Section 2.7.1], it was decided that such methods are unsuitable for probe path planning. The major drawback is that most algorithms which found a solution limit the number of points that can be considered and this would present a problem for an inspection system which needs to consider components consisting of a broad range of sizes and features.

Heuristic algorithms for solving the travelling salesman problem would be more suitable, being capable of handling a larger number of points. However, these are computationally slow, and do not offer any guarantee of yielding a path which is any more efficient than, say, one which is obtained by nearest-neighbour methods.

One other drawback of using travelling salesman algorithms is that the costs associated with moving from one offset point to all other offset points needs to be known in advance. For the cost function described in Section 6.2.1 it is necessary to determine if motion between pairs of points are collision-free and for the scheme of collision detection described in this chapter, this will involve ray-casting from each point to all others, a computationally expensive operation in itself requiring $\frac{n(n-1)}{2}$ rays to be cast. Since heuristic algorithms would then be applied to yield an approximate solution, it is difficult to see the benefits of such algorithms in the case of probe path planning.

In view of this, a nearest-neighbour approach is used whilst not generating the most efficient path does produce an adequate path. The use of attributes to attach extra information to features in the component (see Chapter 7) can be used to influence how the path is constructed. For example, an attribute can be attached which groups half spaces together and hence points which lie on the half spaces. Then when determining a time-efficient path, the grouping can be used to force the grouped points to be inspected together.

6.2.3 Generating An Initial Time-efficient Probe Path

An algorithm has been implemented for deriving a time-efficient probe path. The algorithm uses collision detection techniques based on model growing and ray-casting techniques described in Chapter 4. The model which has been grown for the spherical probe tip is used to generate a path through points which are offset from the generated surface points.

As the model is built, information in the form of attributes is attached to the half spaces and set theoretic operators. These attributes, for example, include the grid spacing used for the point generation algorithms (see Chapter 7 for more information). The method enables such attributes to be attached to geometric entities within the model as they are constructed, and thus allow a *grouping* of those half spaces which are pertinent to a particular entity. Each surface point lies on a particular half space and the grouping of the half spaces defines a corresponding grouping of surface points. The same grouping can be applied to the offset points required for the path planning stage.

The algorithm considers each offset point in turn. The points are first ordered on their x-coordinate and y-coordinate, and the first path point is chosen to be the point with the smallest x value. This choice is arbitrary but does ensure that the path begins at one side of the component.

The next point chosen has to fill two criteria. Firstly, it has to be within the same group as the previous point. Secondly, it must have the minimum cost function value of all remaining points in the group.

The costs are evaluated for moving from the current point to the remaining points in the current group, and the point which has minimum cost is chosen. If there are no more points within the current group, the point with the minimum cost of all remaining points is chosen and the group to which this point belongs becomes the next group.

The algorithm creates a list of path steps representing the probe path. Where collisions occur, obstacles are avoided using the three-step safe-plane path segment described in Section 6.2.1. The path generated in this way does not yet avoid collisions which occur between the non-spherical segments of the probe, and this is the task of the next process.

6.3 The Detection and Avoidance of Collisions

6.3.1 Method of Collision Detection for Facetted Models

The algorithm for detection of collisions for facetted models used the same techniques as those described in Chapter 5 for determining whether a point can be reached by the probe. The probe is modelled as three segments, two vertical cylinders and a sphere, and three grown models were created, one for each of the probe segments. The initial probe path was generated using the model grown for the sphere and so already avoided any unwanted collisions which occurred during motion of the probe tip (see Section 6.2.3). The path has next to be checked for collision between either of the two cylinders and the model.

The path is first tested for the thin cylinder segment representing the probe arm which supports the spherical probe tip. The model is grown such that the cylindrical segment is reduced to the centroid point lying on the axis. Thus the path has to be checked for the points through which this centroid passes as the probe moves, and these are calculated from the original offset points. Collisions are detected by ray casting along the path which the centroid of the cylinder follows as the initial path is followed.

Once the probe arm has been tested, the path is finally checked for the wide cylinder. Again the model is grown so that the centroid is considered, and the path through which this centroid passes is checked using ray casting methods. If collisions are detected for either of the cylindrical parts of the probe then the three-step safe-plane path segment is inserted as when generating the initial probe path for the spherical probe part.

All three segments of the probe have to be considered since detection of a successful path for one segment does not give an indication of the success of the path for the other segments. However, since all points can be reached from the safe-plane, and because the probe arm is thin, if there are no collisions between the spherical probe and the component, then there will often be no collisions between the other parts of the probe. This occurs because clearance of the probe tip from the safe-plane will often imply clearance of the probe arm from the safe plane since the cylindrical arm has smaller radius. However, there are still cases where the spherical tip would be clear and the probe arm would not be [Fig 6.2].

The order of checking of segments is arbitrary since the three segments again are treated independently. There will, in general, be a different path for each ordering since each test could lead to path modification with the insertion of safe-plane path segments. Which ordering is best for reducing the number of modifications is not clear, and perhaps is a topic for future work.

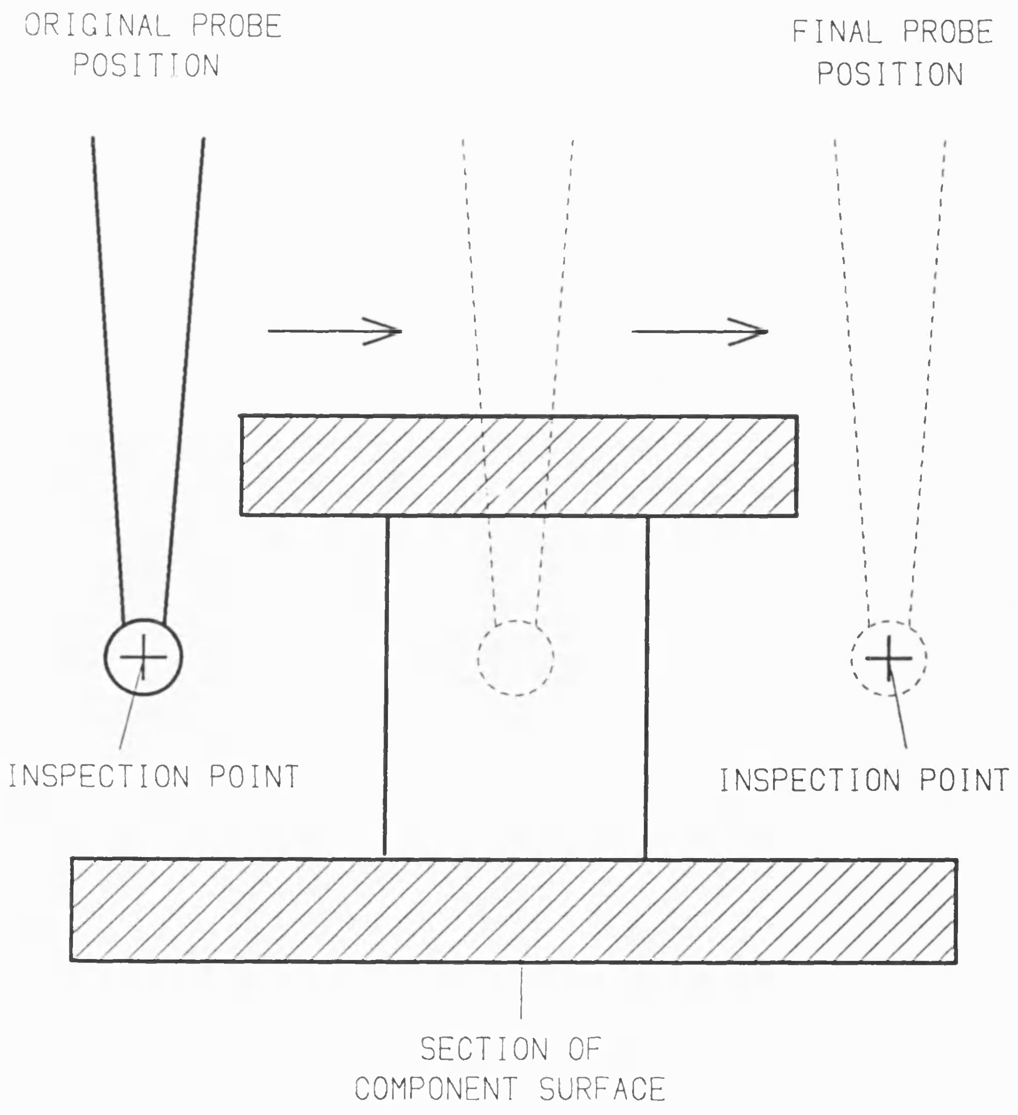


Figure 6.2 Path which is collision-free for the probe tip but not for the probe arm.

6.3.2 Collision Avoidance for Facetted Models

So far, collisions which have been detected have been avoided by inserting a safe-plane path segment into the path. This extra path segment forces the probe to move up to a plane known to be clear of the component and then drives the probe to the second point via this plane. Such an approach may result in unnecessarily large probe movements to the safe-plane to avoid collisions which may be avoided with smaller translations [Fig 6.1].

A more efficient algorithm for avoiding collisions would use the set theoretic model description to produce a path segment which drives the probe around the obstruction. In this way, a move-around path is developed which depends on the component geometry in the locality of the collision point. There are often many paths that could be taken to avoid a collision occurring between the probe and the component when moving between two offset points. An algorithm has been implemented which uses recursive trial and error methods.

The algorithm considers the start and end of each safe-plane path segment within the generated path, together with the point of collision, or hit. The model which corresponds to the probe part for which the collision was detected is used to generate a collision avoidance path segment in the following way.

Four perpendicular test rays lying in the hit plane are cast from the hit point [Fig 6.3(a)] into the model. For each ray, the point at which the ray first leaves solid is found. From the resulting list of boundary points, the point closest to the destination point is chosen. A new path point is inserted into the probe path which is offset from this point, in the direction normal to the hit plane. [Fig 6.3(b)]. A test is made to check that this new path point can be reached directly from the previous path point. If not, an alternative point is chosen from the remaining ray intersection points. If all intersection points fail to be reachable then the safe-plane path segment is used.

If a suitable point is found, then this new path point is then used as a new start point. The above algorithm is then repeated until a clear path exists to the destination point or the obstruction is deemed too difficult or too expensive to avoid, in which case the safe-plane path segment is used.

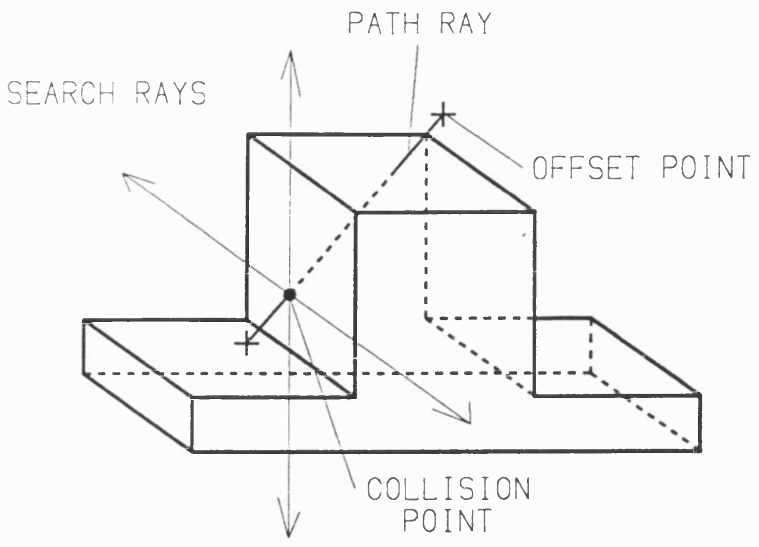


Figure 6.3(a) Generating search rays from collision point between path points.

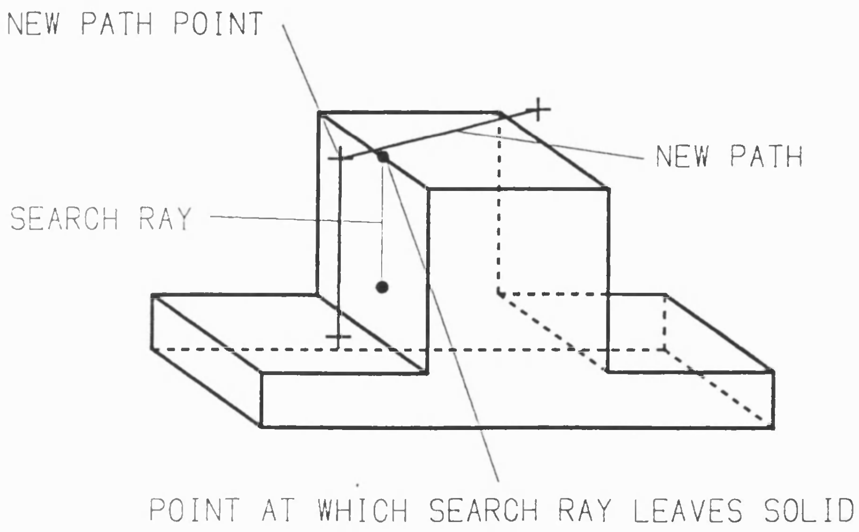


Figure 6.3(b) Collision avoidance using search rays.

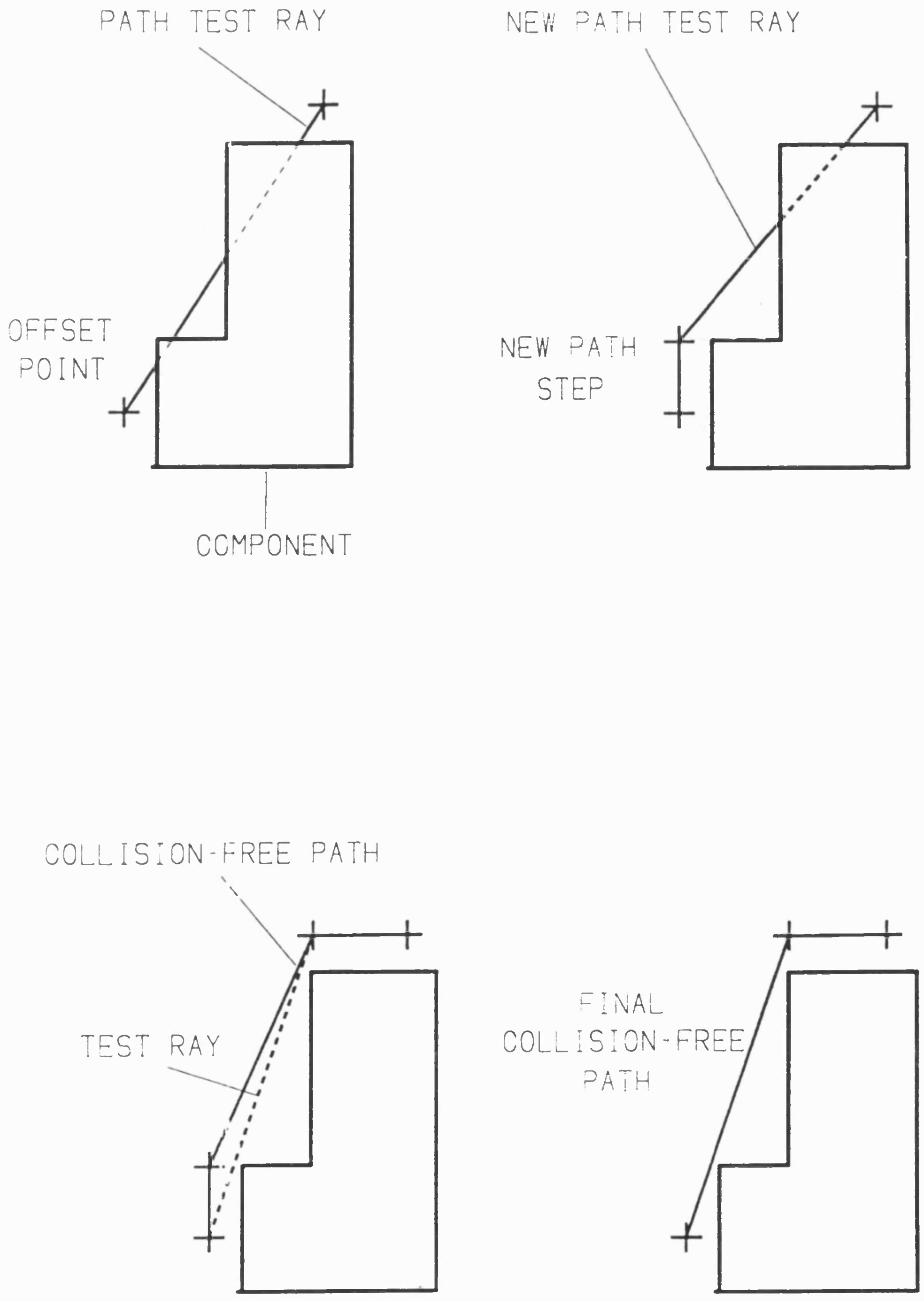


Figure 6.4 Avoiding unwanted path steps using test rays

As each new path step is generated, it is tested for both accessibility from the previous step and from the original first point. In this way, redundant steps can be avoided. If the new step cannot be reached directly from the original start point, then the start point can be updated to be the next point in the new path segment and the test redone until a step which is reachable is found [Fig 6.4].

6.3.3 Examples

Plate 6.1 shows the path for a test piece consisting of blocks and cylinders. Plate 6.2 demonstrates how the probe moves around obstructions. The red path is obstructed since the probe head will collide with the protruding block. The green path shows the corrected path using the described search-ray algorithm.

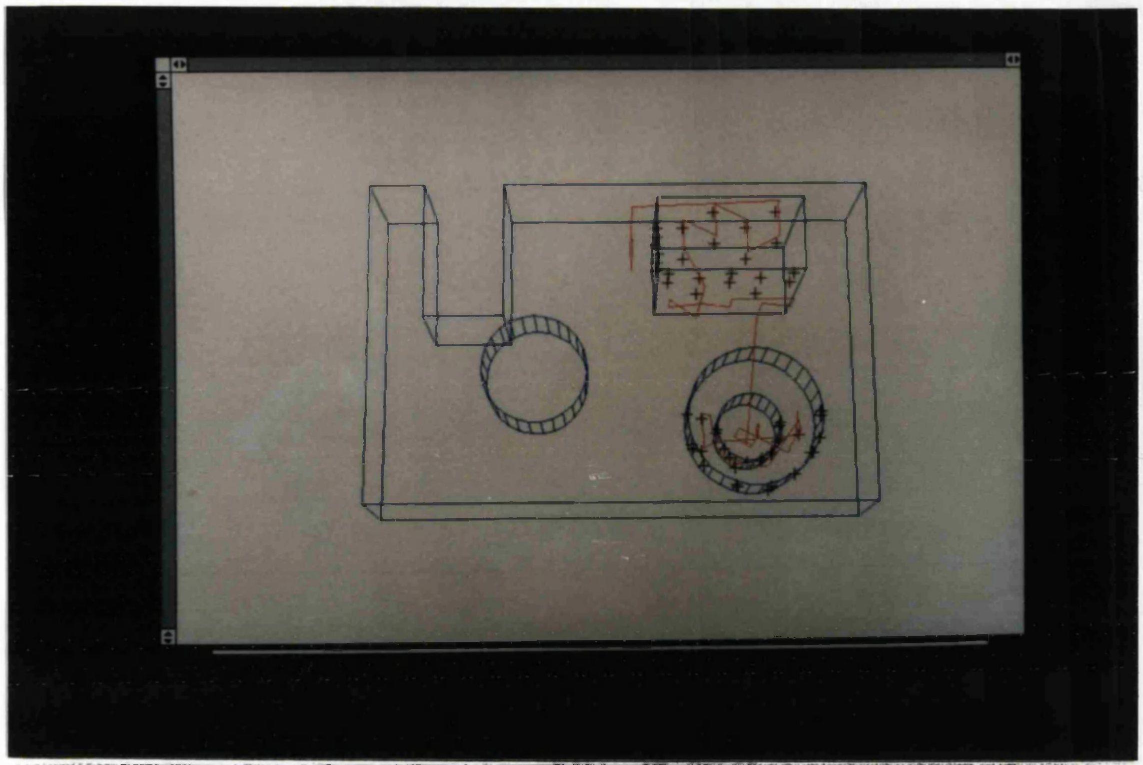


Plate 6.1 Path generated for test piece features and stepped cylindrical hole.

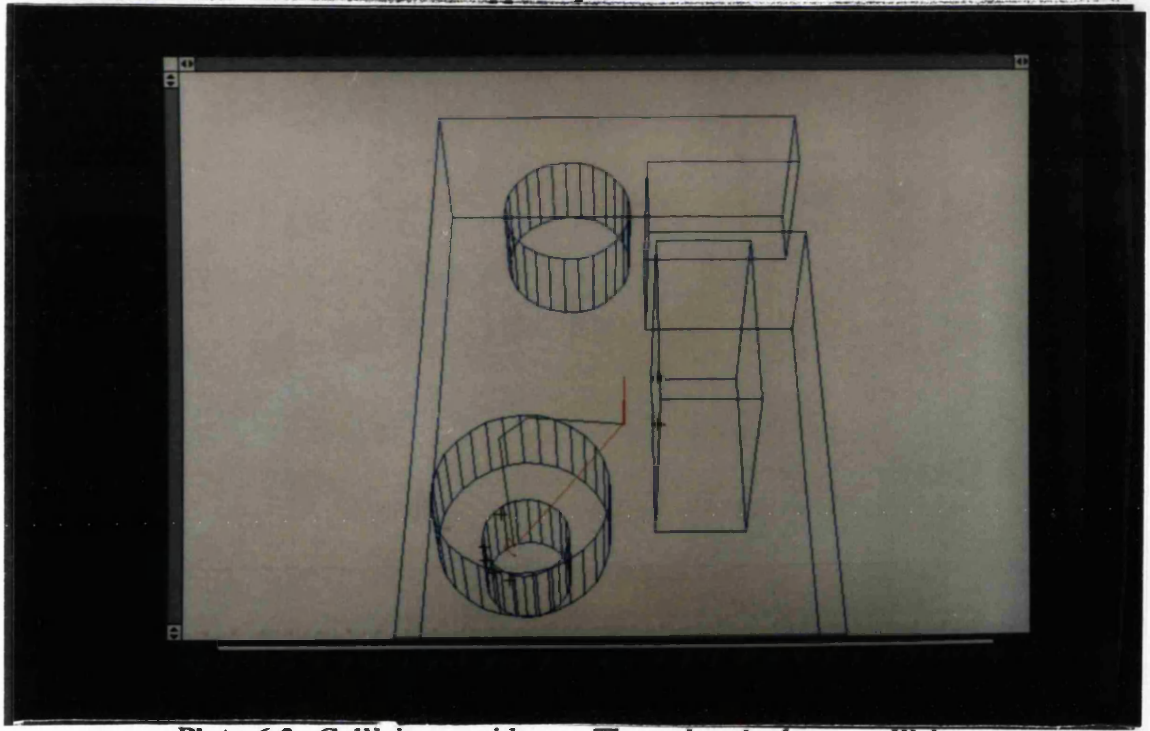


Plate 6.2 Collision avoidance. The red path shows collision steps and the green path shows the corrected path.

6.4 Path Planning for Alternative Probes

The algorithms developed so far are concerned with one particular type of probe consisting of two vertical cylinders and a spherical probe tip. All probes have spherical tips, but some have different probe arms and more degrees of freedom. Examples of these are the cranked probe, where the probe arm consists of two cylinders, one vertical and one horizontal, and the *swivel probe* where the probe arm can be positioned at a number of different orientations.

Assuming the existence of collision detection and avoidance algorithms, the time-efficient algorithms can be applied as they stand, although for the swivel probe extra parameters may be needed to calculate the costs associated with obtaining specific orientations of the probe arm. Hence, the cost function will no longer only depend on the distance between the two points but also on the various orientation transformations which occur in moving from one point to the other. The cranked probe requires no modification of the cost function since motion is still purely translational.

The swivel probe introduces a new problem to path planning. With each path point, there is now a required orientation and so the path segments will have also to list the sequence of orientations required in moving from one point to another. Such a path is non unique since there will be many ways of obtaining the desired orientation. For example, the probe could move an arbitrary distance along the line joining the two points, rotate until the desired orientation is obtained and then continue along the line until at the second point. Clearly this defines an infinite number of possibilities. Thus a method is required for generating a sequence of steps from one point to another. Further, a search has to be made for a collision-free solution and, to accommodate cases when such a solution does not exist, a criterion has to be used to decide when enough possibilities have been tried.

The safe-plane path segment can still be used to avoid collisions as they occur. The collision detection algorithms for the cranked probe are the same as those for the simple vertical probe. If a safe-plane path segment has to be added, the considerations mentioned in Chapter 5 for cranked probes need to be used in determining how to move to and from the safe-plane.

However, the swivel probe requires a different approach since the linear translation of each different orientated probe arm requires a different grown model. Path-planning now requires algorithms similar to those used in robot path-planning for determining a collision-free sequence of rotations and translations between points. The grown model would be useful in the detection of collisions in such algorithms.

For swivel probes, the loci of the probe tip and arm are no longer linear since during orientation changes they will move along an arc. Ray-casting along the arcs is one solution, although intersection calculations will now involve non-linear equations. These present no difficulty so long as the probe is stationary when being rotated in which case the probe loci is circular. However, new complexities are introduced if the probe is swivelled and translated at the same time. In this case arcs could be approximated as linear segments and the usual ray-casting methods used.

In summary, cranked probes require little or no changes to the path generation algorithms. Swivel probes present new problems as a result of the extra degrees of freedom and non-linear path segments.

This is also true in avoiding collisions and in improving safe-plane path segments. The cranked probe can be successfully navigated around obstructions using the same algorithms as described above, but the swivel probes need special procedures to take into account the possible orientations.

6.6 Conclusions

This chapter has described various algorithms used for path planning of a simple measuring probe, modelled by two vertical cylinders and a spherical probe tip. The algorithms have been applied to examples of various complexity and successfully generated collision-free measuring paths which have been subsequently used to collect measurements.

Work on collision detection using solid models has been considered in several other projects [Section 2.8.2]. The approach used here is new in that it uses a set theoretic solid model to calculate the free-space available to the probe by growing the obstacle, i.e the component.

Extending the algorithms to curved-surfaces and alternative probes creates various problems. To solve these requires solving the collision-detection problem for more complex modelling in the form of curved-surfaces, and for probes with different degrees of freedom. The techniques used in the algorithms described in this chapter involve growing models and ray-casting, and this restricts consideration to facetted models. However, the basic approach of generating the path for the spherical probe tip and correcting it where collisions for the other probe segments occur is a valid and workable solution for both facetted and curved surface models.

CHAPTER 7

ASSIGNING ATTRIBUTES TO SOLID MODELS

7.1 Introduction

In many engineering applications of solid modelling, non-geometrical information about components is required in addition to their geometric description. For example, the colour of surfaces is necessary for picture generation and the density of material is required to calculate mass. Certain features may need to be tagged with names for later reference in the same way datums are labelled in engineering drawings.

From an inspection point of view, parameters are required for the point generation and the probe-path determination algorithms. Further, geometric tolerances, consisting of non-geometrical information such as the tolerance type and tolerance width, are required for the analysis of measurements.

This chapter describes a method of attaching non-geometrical information to set theoretic solid models. The technique used involves assigning attributes to a set theoretic expression. (The effect of this is to introduce a new type of operator, the dyadic *attribute* operator, which assigns a list of attributes to an operand which is a set theoretic expression). Attributes can be assigned to groups of half spaces which define geometric entities, (i.e. axes, planar surfaces, spheres, cylinders, cones) of the component.

The technique has been implemented for a solid modeller developed at Bath University which uses a model description language to build the models. However, the choice of attribute functions and techniques used to implement them are applicable to any model description language. The first part of this chapter describes how the language definition and its compiler have been improved to enable attributes as described above to be attached to a model as it is built. The second part of the chapter describes some of the types and applications of attributes which can be applied using this method, with particular attention to geometrical tolerancing.

7.2 The Existing Model Description Language and its Compiler

The modeller uses planar half spaces as its basic primitive. More complicated primitives can be constructed from these using set theory operators defined within the language. Curved surface half spaces can be constructed by the *arithmetic* combinations of these planar half spaces. For example, if the planar half space A, represented by polynomial $P(x,y,z)$, is orthogonal to the planar half space B, represented by polynomial $Q(x,y,z)$, then a cylinder, radius r, whose axis lies along the line of intersection between A and B is represented by the second-degree polynomial, $S(x,y,z)$, where:

$$S(x,y,z) = (P(x,y,z))^2 + (Q(x,y,z))^2 - r^2$$

The language consists of the usual datatypes, such as integers and reals, but also has some geometric types such as *points*, *lines* and *planes*, and a special type for storing set theoretic expressions, called a *set*. Arithmetic operations can be applied to all datatypes except sets, and set theory operators apply to sets only. In addition, there is a set of geometric functions which apply transformations to geometric datatypes and sets. Examples include the scaling function and the rotation function. A model is built as a single set using the set theoretic combination of other sets.

The procedure **box**, which builds a cuboid given two diagonal corners, is shown in Fig 7.1 to illustrate the syntax of the language. The planar half-spaces which correspond to the cuboid sides are built as sets and the intersection between all the resulting six sets is written into another set.

The language enables planar half-spaces to have a colour attached to them, but this is the only non-geometric information considered. It is attached to sets using a function defined within the language.

A compiler is used to compile and run the source code written in the model description language to create a model description file. This file contains a list of planar half-spaces and an expression, written in reverse-polish order, containing references to the half spaces together with set theoretic operators, constants and arithmetic operators.

The set datatype contains the list of planar half spaces, constants and operators which define the set. A planar half space is stored as four floating point numbers, containing the plane equation coefficients a, b, c, d and an extra field for storing the colour code.

```
;
; Function to build cuboid from opposite corner points
;
        FUNCTION box( minpt:Point, maxpt:Point) :Set

; Variable definitions
        Sets {minxf, minyf, minzf,maxxf, maxyf, maxzf}
        Sets {cuboid}
        Points {x_vec, y_vec, z_vec,mx_vec, my_vec, mz_vec}

        {

; Define normal directions to faces
                x_vec := pt(1,0,0)
                y_vec := pt(0,1,0)
                z_vec := pt(0,0,1)

                mx_vec := - x_vec
                my_vec := -y_vec
                mz_vec := - z_vec

; Define planes (as sets) passing through lowest corner
                minxf := space(mx_vec,minpt)
                minyf := space(my_vec,minpt)
                minzf := space(mz_vec,minpt)

; Define planes (as sets) passing through highest corner
                maxxf := space(x_vec,maxpt)
                maxyf := space(y_vec,maxpt)
                maxzf := space(z_vec,maxpt)

; Intersect faces together to form cube
                cuboid := minxf & minyf & minzf & maxxf &
                          maxyf & maxzf
; Return result
                RETURN(cuboid)
        }
```

Figure 7.1 Sample of model description language.

7.3 Assigning Attributes in The Model Description Language

The language requires extra functions to allow attributes to be assigned. These functions take as their arguments a set variable and the attribute information. The types of attributes needed for attaching geometric tolerances to models are discussed in section 7.4.

Three attribute functions have been implemented. **tol_set** assigns tolerances to the argument set, **datum** labels a set with a text string, and **attribute** assigns a single attribute to the set together with a code which is to allow the user to identify the attribute. The function **attribute** allows any datatype, except sets, to be considered as an attribute for the argument set.

The example in Fig 7.2 shows how the attribute functions are used to label and tolerance the faces of the block shown in Fig 7.1.

The colour field of the half-space representation is now changed to be a pointer to an attribute list, and colour is now considered an attribute of half spaces. The list contains the attribute information together with the length of set theory expression, or *scope*, for which the attribute is valid. In this way, the segment of model to which the attribute applies is identified.

When an attribute is assigned to a set, it is added to the attribute list for the first half space within the reverse-polish list which represents the set theoretic expression of the set. The scope of the attribute is also added to the list. The other half spaces of the set may themselves have attribute lists valid over sub-segments of the set theoretic expression stored in the set.

```
;
; Function to build toleranced cuboid from opposite corner points
;
        FUNCTION tol_box( minpt:Point, maxpt:Point) :Set

; Variable definitions
        Sets {minxf, minyf, minzf,maxxf, maxyf, maxzf}
        Sets {cuboid}
        Points {x_vec, y_vec, z_vec,mx_vec, my_vec, mz_vec}

        {
; Define normal directions to faces

                x_vec := pt(1,0,0)
                y_vec := pt(0,1,0)
                z_vec := pt(0,0,1)

                mx_vec := - x_vec
                my_vec := - y_vec
                mz_vec := - z_vec

; Define planar faces
                minxf := space(mx_vec,minpt)
                minyf := space(my_vec,minpt)
                minzf := space(mz_vec,minpt)

                maxxf := space(x_vec,maxpt)
                maxyf := space(y_vec,maxpt)
                maxzf := space(z_vec,maxpt)

; Label datum reference plane
                maxyf := datum (maxyf,"Y-MAX FACE")

; Tolerance parallel and perpendicular planes
                minyf :=
                  tol_set (maxyf,PARALLELISM,0.5,pt(0,0,0),"Y-MAX FACE)

                maxxf :=
                  tol_set  maxyf,SQUARENESS,0.5,pt(0,0,0),"Y-MAX FACE)

; Calculate and return result
                cuboid := minxf & minyf & minzf & maxxf &
                          maxyf & maxzf

                RETURN(cuboid)
        }
```

Figure 7.2 Example of using attribute functions.

The new compiler creates an attribute file together with the model file. The attribute file is directly related to the model file in that every planar half space has an entry in the attribute file. The scope of the attributes are also written to the attribute file and so the information is dependent on the structure of the reverse-polish expression written to the model file. If this expression is altered, then the scope will no longer be valid.

7.4 Attributes

7.4.1 Review of Geometric Tolerances

Geometric tolerances are assigned to various geometric entities of a component in order to maintain the quality of production. The geometric entities, as defined by BS308 part 3, consist of planes, axes and simple quadrics, namely spheres, cylinders and cones. Some tolerances reference other geometric entities as datums, and these are either planes or axes. The tolerances fall into three categories: form, attitude and location.

Form tolerances are applied to the surface of features and as such are restricted to planes and simple quadrics. They do not reference datums. They define the allowed variation in surface shape and include the tolerances of FLATNESS and ROUNDNESS.

Attitude tolerances constrain the orientation of one geometric entity relative to one or more other geometric entities. The entities that can be constrained by attitude tolerances are planes and axes, including axes of cylinders and cones. The constraint is made relative to datums. Examples are PARALLELISM, SQUARENESS and ANGULARITY. In some cases, depending on the types of the toleranced and datum entities, a direction is required to define the plane within which the constraint holds. Fig 7.3 shows two possible planes of constraints which geometric tolerances can define for an axis of a block.

Location tolerances constrain the relative position of geometric entities. As with attitude tolerances, they are restricted to axes and planes, and also reference datums. Again, depending on the type of toleranced and datum entities, a direction may be required to define the plane of constraint. Examples of location tolerance are POSITION, SYMMETRY and CONCENTRICITY.

The interpretation of geometric tolerances involves using the tolerance information to determine a *tolerance zone*. This may be a planar area, or a planar or cylindrical volume. In most cases, the information required to specify the tolerance zone can be determined from the toleranced entity and datum entity types. One exception is planar area tolerance zones which require extra information in the form of a direction to define the plane within which the area applies [Fig 7.3]. Another exception occurs when axes are toleranced. Although the planar volume zone type for axes can be determined from the types of the toleranced and datum entities, extra information is required to distinguish between axes zone types which are planar areas and cylindrical volumes. Chapter 8 describes the relationship between entities and zones in greater detail.

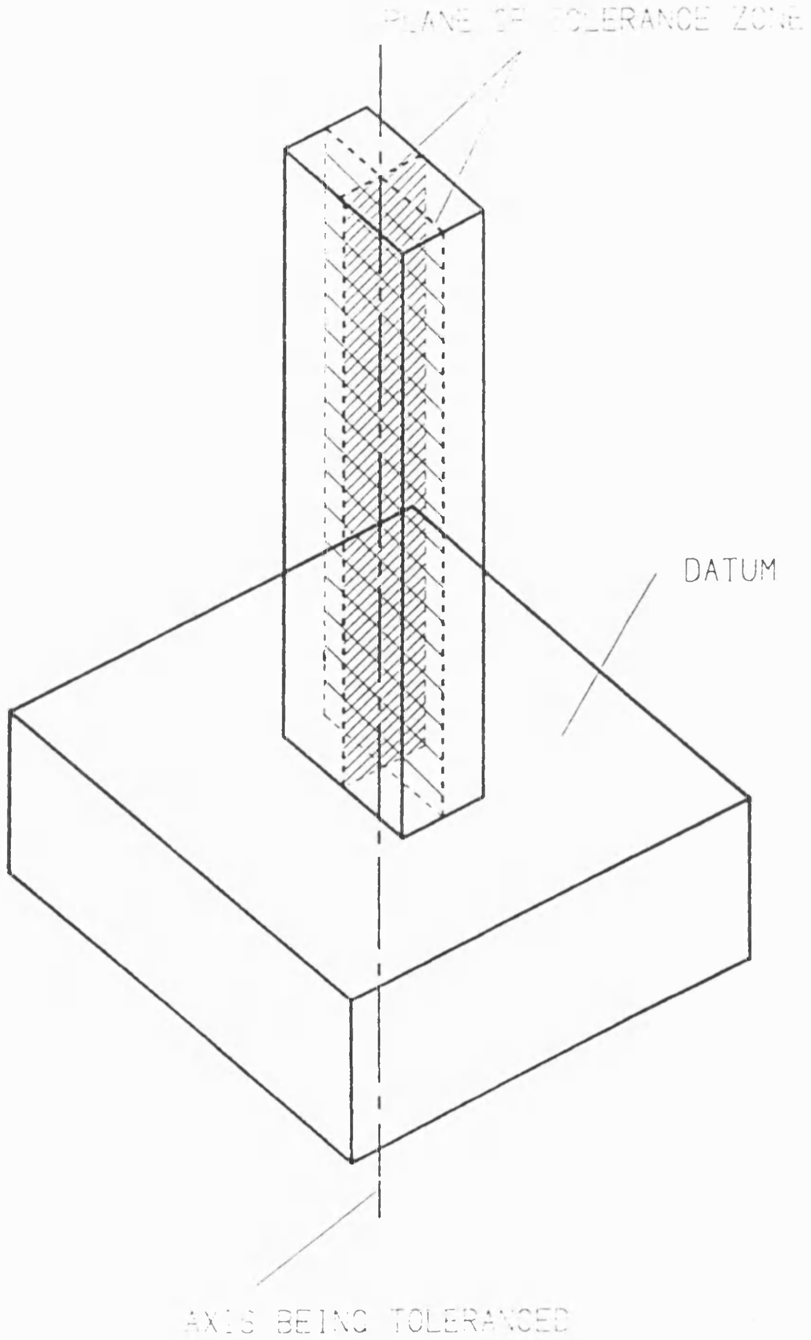
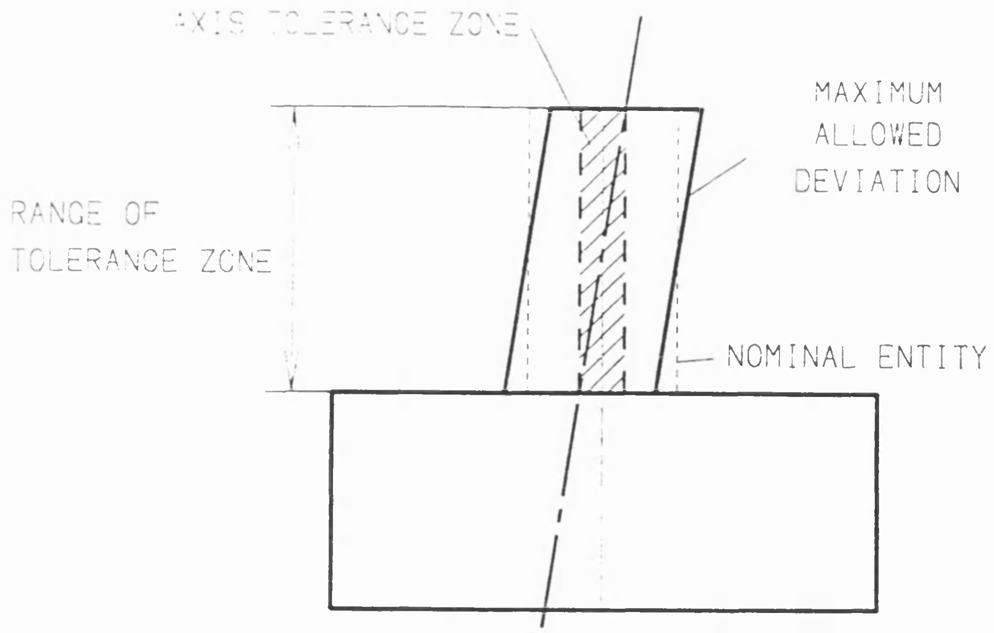


Figure 7.3 Two alternative tolerance planes for axis. A direction field is needed to select the required plane.

Checking the tolerance involves testing that the measured geometric entity lies within the given zone. However, the extent of the entity to which the zone is to be applied, or the *range* of the tolerance zone, is not defined. To see how this effects interpretation of tolerances see Fig 7.4 which illustrates how increasing the range of a planar area tolerance applied to an axis effectively reduces the allowed deviation of the axis.

No information is given about this range of tolerance zone in the British Standard BS308 Part 3 on geometric tolerances which implies that the range may be deduced from the engineering drawing. Whilst such a deduction is possible by a human inspector working from a drawing, it does not form the basis for an automatic approach based on solid modelling. The necessary information can be extracted from the model by assuming that the toleranced entity is bounded, and that the bounds of the entity define the range of the zone. Chapter 8 describes how this is done.

The table 7.1 describes the tolerances, zones and other requirements for the various geometric tolerances considered. The standard BS308 Part 3 and Chapter 8 of this thesis should be consulted for further information.



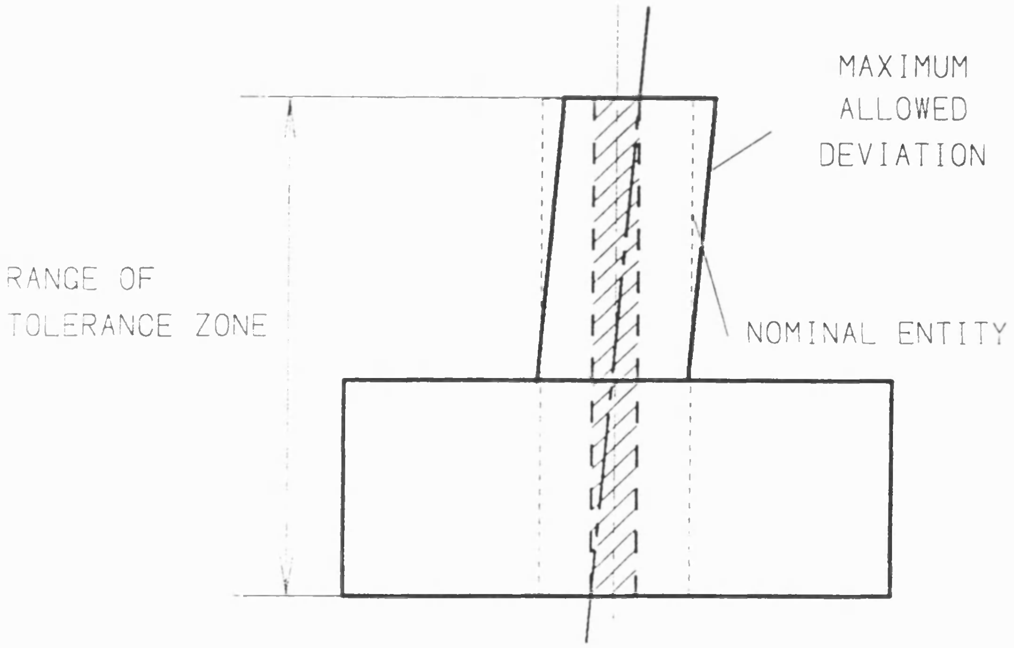


Figure 7.4 The effects of different ranges on tolerance zones. The longer the range effectively increases the constraint on allowed deviations.

Form Tolerances			
Tolerance	**Features**	**Datums**	**Direction**
Flatness	Planes Only	No Datums	No Direction
Roundness	Spheres	No Datums	No Direction
	Cylinders		
	Cones		
Cylindricity	Cylinders Only	No Datums	No View Direction

Table 7.1(a) Tolerances of FORM.

Attitude Tolerances			
Tolerance	**Features**	**Datums**	**Direction**
Squareness	Axis	Axis	Planar Direction
	Axis	Plane	Optional Direction
	Plane	Line	No Direction
	Plane	Plane	No Direction
Parallelism	Axis	Axis	Optional Direction
	Axis	Plane	No Direction
	Plane	Axis	No Direction
	Plane	Plane	No Direction
Angularity	Axis	Axis	Planar Direction
	Axis	Plane	Optional Direction
	Plane	Axis	No Direction
	Plane	Plane	No Direction

Table 7.1(b) Tolerances of ATTITUDE.

Location Tolerances			
Tolerance	**Features**	**Datums**	**Direction**
Position	Axis	Axis	Planar Direction
	Axis	Plane	Planar Direction
	Plane	Axis	No Direction
	Plane	Plane	No Direction
Concentricity	Axis Only	Axis Only	No Direction
Symmetry	Axis	Two Axes	Planar Direction
	Axis	Two Planes	Optional Direction
	Plane	Two Planes	No Direction

Table 7.1(c) Tolerances of LOCATION.

7.4.2 Tolerance and Datum Attributes

The description of geometric tolerances above suggests an attribute containing four pieces of information. A code is required to define the tolerance type and a floating point number is required to define the tolerance value, or tolerance width. For planar area tolerances, a direction is required to define the plane within which the area applies. For tolerances which reference datums, a method of defining the referenced entity is required. The method chosen is to allow the user to attach a name to such datum entities (as an attribute), and then use this name to reference them. Thus a text string is required to identify the datums. (The string field can also be used to define any other information for the tolerance; for example, to define whether a maximum-material constraint should be applied)

Geometric entities are toleranced using the function **tol_set** which has the four necessary arguments in addition to the set argument. The function is applied to the bounded entities as they are defined. Transformations on sets and arithmetic operations on half spaces will remove tolerances from the set and thus tolerances should only be applied to the entity when it has been correctly located, orientated and is of the correct size.

When no direction is needed for tolerance interpretation, a direction of zero length is given. This is used to decide between planar area tolerance zones and cylindrical volume tolerance zones when applied to axes (see Section 7.4.1); if a non-zero direction is present, then a planar area zone is assumed. (see Chapter 8).

Datums are defined using the function **datum** which has one argument, the datum name, in addition to the set argument. Tolerances can only reference entities labelled using this function. Since the labels do not depend on the geometry of the entity, they will survive transformation and arithmetic operations on sets.

7.4.3 General Attributes

General attributes may be applied to models using the **attribute** function. This function allows any non-set single data type such as integers, floating point numbers, points and strings to be assigned to sets, together with a user defined code which is used to specify the attribute type.

The function has three arguments in addition to the set argument; the user-defined code, the attribute information and a flag to determine how the attribute propagates across geometric transformations. This is necessary because some attributes, such as geometric tolerances, become invalid if the entity to which they apply is subject to geometric transformations. These attributes will be discarded during such transformations. Others, such as colour, are independent of the geometry of the entity and so will be retained during transformations.

General attributes allow various types of non-geometric information, such as material type or density, to be assigned to sets, and, in principle, the tolerance and datum attributes can be assigned using general attributes. Each tolerance argument could be applied as a separate attribute and hence a tolerance can have an arbitrary number of fields. Similarly datums could be applied as a general attribute with a special code to define it as a datum required specifically for the tolerances.

To summarise, the definition of general attributes can allow the definition of more specialised attributes with any number of arguments.

7.5 Using Attributes for Inspection Planning

Tolerance (and datum) attributes are assigned to entities in order to specify a quality of production by applying constraints to the allowed deviations of the shape and size of entities. Geometric tolerances are readily represented as attributes of solid models since they apply to 3-Dimensional geometric entities. Dimensional tolerances in contrast are applied to 2-Dimensional entities such as planar distances and radii. A different method of attaching attributes to solid models would be required to include such tolerances.

General attributes are used at the inspection planning stage to define parameters required by the point generation and path generation algorithms. These include the grid spacing for points (Chapter 3) and the half space grouping information for path generation (Chapter 6).

The attributes are also used to label planes required as measuring datums for coordinate system alignment during measuring. The CMM used for measuring the component will usually have a different coordinate system to the model and, in order to align the two systems, a transformation matrix is calculated from the results of probing the labelled measure datum planes (see Appendix 1).

Chapter 8 describes how attributes can be used in the identification of geometric entities given its set theoretic description and also in the extraction of specific geometric parameters, such as radii and axial directions, from the same expression. This is useful during the analysis of inspection results.

7.6 Conclusions

The problem of assigning attributes and tolerances to solid models has been considered by various authors. This chapter describes a method which attaches such non-geometric information to sub-segments of set theoretic expression within a solid model. The approach is similar to that used by A. Requicha [see Section 2.9], but by restricting attention to geometric tolerances there is no need for the face labelling scheme.

The techniques described in this chapter offer a method of representing geometric tolerances in set theoretic solid models. The method inherits the advantages of set theoretic solid models, such as simplicity in description, but also inherits the disadvantage of being difficult to change once built. If a new geometric entity is added or an old entity is removed, then the set theoretic expression will change and the range of influence of attributes may become invalid. In general, the model has to be re-compiled every time a change is made and this makes interactive changes difficult.

The ability to assign general attributes has several applications within automated inspection, and clearly can be used for other purposes within manufacturing.

CHAPTER 8

ANALYSIS OF MEASURED DATA

8.1 Introduction

This chapter describes methods of processing the results of coordinate inspection to check that the measured features of the component lie within specified geometric tolerances. This is done using a set theoretic solid model of the component which has appropriate geometric tolerances assigned as attributes using the techniques described in Chapter 7. The techniques have been developed for faceted solid models, where half spaces consist of planes only, but indication is given where possible to show how they can be applied to curved half spaces.

The first part of this chapter describes how geometric information is extracted from both the solid model and from the measurement results. This information is needed for the algorithms which check that the component shape lies within the specified geometric tolerances. The type of information required depends on the type of tolerance and the geometric entity (i.e. axis, plane, cylinder, cone or sphere) within the component to which the tolerance applies.

Algorithms which check that the geometric entities lie within the specified tolerance have been developed. The algorithms apply the idea of *tolerance zones* discussed in the British Standard on geometric tolerances BS308 Part 3. How these zones are constructed is described in the second part of this chapter.

Finally the algorithms which check the tolerances are described. These algorithms compare the geometric information extracted from the measured results with the corresponding nominal information in the solid model. Tolerance zones are used to decide whether any discrepancies between the two are within the specified geometric tolerances.

8.2 Extraction of Information About Geometric Entities

8.2.1 Introduction

BS308 part 3 describes the entities to which geometric tolerances can be applied. The work in this chapter considers axes, planes, cylinders, cones and spheres. Chapter 7 provided an overview of the tolerances which are applied to these entities, and identified the following: FLATNESS, ROUNDNESS, CYLINDRICITY, POSITION, CONCENTRICITY, SYMMETRY, SQUARENESS, PARALLELISM and ANGULARITY.

The tolerances constrain parameters which define the size and position of the entities. The parameters constrained depends on the entity and the tolerance. The following lists all the parameters which may be constrained for each entity.

(a) For an axis, they consist of the axis direction and location

(b) For a plane, they consist of the normal direction and position.

(c) For a sphere, they consist of the location of the centre and the radius.

(d) For a cylinder, they consist of the axial information specified in (a) together with the radius

(e) For a cone, they consist of the axial direction, the position of the apex and the cone half-angle

This geometric parameter information is not explicitly available in the solid model, and the next section describes techniques for extracting it.

When a component is inspected a set of measured points is obtained. These points have to be processed to extract the parameters for comparison with the corresponding expected or nominal parameters contained in the solid model and Section 8.2.3 describes techniques for doing this.

8.2.2 Extracting Geometric Information from the Solid Model

The solid model consists of a set theoretic expression containing half spaces and set theory operators. As described in Chapter 7, geometric tolerances are applied to geometric entities within the component by attaching tolerance attributes to the segment of set theory expression which defines the entity.

The set theoretic expression completely defines the geometric entity and in principle it should be possible to both identify the entity and extract the required geometric parameters. In practice both of these problems are difficult.

For facetted models, algorithms for identification of the quadric entity types (i.e. whether they are a cone, cylinder or sphere) become complex. Because of this, attributes are used to attach "labels" to the set theory expressions which represent the geometric entities and thus enable identification.

Models which use curved surface half spaces will not have this problem since it is possible to identify the simple quadrics (sphere, cone and cylinder) from an implicit polynomial equation. Methods exist (see Section 2.10) which use eigen values and eigen vectors to transform the half space polynomial to a canonical polynomial. Classification of the quadric can be made by considering the coefficients of this polynomial [Table 8.1]. The canonical form also enables the geometric parameters to be extracted, again by considering the coefficients. Difficulties arise with identification if the original polynomial defining the quadric is ill-conditioned, i.e. if the difference in magnitudes between the coefficients is large. When this occurs, the coefficients within the canonical polynomial become sensitive to floating point inaccuracies which arise during the transformation from the original polynomial, and this leads to incorrect identification of the simple quadric. However, by using the same scheme of attributes as that used for facetted models this problem of incorrect quadric identification through ill-conditioning is avoided.

Once the type of geometrical entity type is known, the parameter information can be extracted. Again this is easier for curved surfaces, which can use eigen value techniques, although ill-conditioning still produces problems. For facetted models, assumptions have to be made about the way the entity has been modelled. (The non-uniqueness of representations within solid models was discussed in Chapter 2). Alternatively, the geometric parameter information can also be attached as attributes to the set theory expression describing the entity. This makes extraction of parameter information quicker at a cost of introducing redundant information.

Equation	Coefficients Class	Quadric Type
$ax^2 + by^2 + z^2$	$a > 0, b > 0, c > 0$	Point
	$a > 0, b > 0, c < 0$	Cone
$ax^2 + by^2 + cz^2 + d$	$a > 0, b > 0, c > 0, d < 0$ $a = b = c$	Sphere
	$a > 0, b > 0, c > 0, d < 0$ $a \neq b \neq c$	Ellipsoid
	$a > 0, b > 0, c < 0, d < 0$	One-Sheet Hyperboloid
	$a > 0, b < 0, c < 0, d < 0$	Two-Sheet Hyperboloid
$ax^2 + by^2 + d$	$a > 0, b > 0, d < 0$ $a = b$	Cylinder
	$a > 0, b > 0, d < 0$ $a \neq b$	Elliptic Cylinder
	$a > 0, b < 0, d < 0$	Hyperbolic Cylinder
$ax^2 + by^2 + 2wz$	$a > 0, b > 0, w < 0$	Elliptic Paraboloid
	$a > 0, b < 0, w < 0$	Hyperbolic Paraboloid
$ax^2 + 2wz$	$a > 0, w \neq 0$	Parabolic Cylinder

Table 8.1 Identification of quadrics from second-degree polynomials.

Tolerance checking algorithms described in this thesis use attributes as the mechanism for obtaining geometric parameter information from solid models.

8.2.3 Extracting Geometric Information From Measurements

For geometric tolerances which apply to surfaces, such as FLATNESS and ROUNDNESS, the measured points themselves can be used directly to test the tolerance. However, for checking all other geometric tolerances, information is required about the geometric parameters within the measured entities.

If the entity type is known in advance, algorithms exist for extracting the required parameters for planes, cylinders, spheres and cones.

One general method is to fit, using constrained least squares or eigen vector methods, a general quadric to the measured points which has the form:

$$P = ax^2 + by^2 + cz^2 + 2eyz + 2fxz + 2gxy + 2ux + 2vy + 2wz + d$$

Then using the coefficients, the required information can be derived using similar techniques to those described in Section 8.2.2. In principle, this method also enables the type of entity which the measured points best represent to be determined. In practice, the coefficients are nearly always ill-conditioned and because of floating point inaccuracies in their calculation, this leads to incorrect conclusions about entity types and also further inaccuracies in calculation of the corresponding geometric parameters.

In view of this, algorithms developed at the National Physical Laboratory (NPL) which use constrained least squares methods to estimate directly the entity parameters from the measured point data are used. The methods are iterative procedures based on Gauss-Newton search methods, and different procedures are used for different entities. The nominal parameters, i.e. those extracted from the corresponding entities within the solid model, are used as the initial conditions for iteration.

Using these NPL algorithms, the parameter information required for the various geometric tolerance tests are extracted from the measured point set. Fig 8.4 and Appendix 2 show examples of the use of these algorithms.

8.3 Tolerance Zones

8.3.1 Zone Characteristics and Derivation

For tolerances which require the comparisons of measured entities with nominal entities, a tolerance zone is used. The tolerance zone is the region within which the measured entity should lie. There are three types of zone defined by British Standard BS308 Part 3; *planar area, planar volume* and *cylindrical volume.* The type of zone depends on the tolerance type, the type of entity and the type of any datum entities which are referenced. As explained in Chapter 7, extra information in the form of a viewing direction is also required in some cases to completely specify the zone if it is a planar area type. This direction defines the normal to the plane in which the area zone lies.

The relationship between the zone and tolerances, features and datums is summarised in Table 8.2. (BS308 should be consulted for more details). Generally, zones defined by cylindrical volumes or planar areas are used to check axes, and planar volumes are used to check planes. The datum feature is used mostly to specify the orientation of the zone, although for some cases it also specifies the zone type. For example, PARALLELISM of an axis with respect to a datum plane requires a planar volume, SQUARENESS of an axis with respect to a datum plane requires a planar area or cylindrical volume. If there is a choice between planar area or cylindrical volume for an axis, the existence of a viewing direction in the list of attributes which specify the tolerance is assumed to imply that a planar area zone is required.

The width of the planar zone, or radius of cylindrical zone, is determined from the tolerance value. The position of the zone depends on whether the tolerance is a location or attitude type (see Chapter 7 for types of Geometric Tolerances). If it is a location, then the position is fixed relative to some datum. If an attitude, then the position is not relevant.

In summary, the tolerance type, toleranced entity type and the datum entity type are considered in determining the tolerance zone. Extra information is obtained from the model in the form of attributes, such as the viewing direction, or from the set theoretic expression defining the entity. The latter case includes information about the extent or *range* of the tolerance zone, and this is the topic of the next section.

Form Tolerances		
Tolerance	**Entity**	**Zone Type**
Flatness	Plane Only	Planar Volume
Roundness	Sphere Only	Spherical Volume
	Sphere and Direction	Planar Area
	Cylinder Only	Cylindrical Volume
	Cones Only	Conical Volume
Cylindricity	Cylinders Only	Cylindrical Volume

Table 8.2(a) Zone types for tolerances of FORM.

Attitude Tolerances			
Tolerance	**Entity**	**Datums**	**Zone Type**
Squareness	Axis	Axis	Planar Direction
	Axis and Direction	Plane	Planar Area
	Axis Only	Plane	Cylindrical Volume
	Plane	Line	Planar Volume
	Plane	Plane	Planar Volume
Parallelism	Axis and Direction	Axis	Planar Area
	Axis Only	Axis	Cylindrical Volume
	Axis	Plane	Planar Volume
	Plane	Axis	Planar Volume
	Plane	Plane	Planar Volume
Angularity	Axis	Axis	Planar Area
	Axis and Direction	Plane	Planar Area
	Axis Only	Plane	Cylindrical Volume
	Plane	Axis	Planar Volume
	Plane	Plane	Planar Volume

Table 8.2(b) Zone types for tolerances of ATTITUDE.

Location Tolerances			
Tolerance	**Entity**	**Datums**	**Zone Type**
Position	Axis Only	Axis	Cylindrical Volume
	Axis and Direction	Axis	Planar Area
	Axis Only	Axis	Cylindrical Volume
	Axis and Direction	Plane	Planar Area
	Plane	Axis	Planar Volume
	Plane	Plane	Planar Volume
Concentricity	Axis Only	Axis Only	Cylindrical Volume
Symmetry	Axis and Direction	Two Axes	Planar Area
	Axis and Direction	Two Planes	Planar Volume
	Axis Only	Two Planes	Cylindrical Volume
	Plane Only	Two Planes	Planar Volume

Table 8.2(c) Zone types for tolerances of LOCATION.

8.3.3 Range of Tolerance Zones

As described in Chapter 7, the range of the tolerance zone is not defined for geometric tolerances as specified in BS308. For example, the extent of planar or cylindrical volumes for axes has not yet been determined. Fig 7.2 shows how important this information is for checking of tolerances.

For this reason, it is assumed that tolerances are applied to bounded entities from which the range can be derived. For example, if an axes of a cylinder is toleranced, then given that the cylinder is bounded, the range of the zone can be determined by casting a ray into the set theoretic description of the cylinder, and calculating the intersection points. These points can then be used to determine the upper and lower bounds on the zone in the direction of the axis.

8.4 Tolerance Checks

8.4.1 Types of Checks

In order to assess whether geometric tolerances are satisfied, the measured point data has to be compared with the toleranced nominal component description for each toleranced entity. The type of checks required depend on the tolerance type and the entity to which it applies.

As mentioned in section 8.2.3, FLATNESS and ROUNDNESS tolerances which constrain the surface variations can be checked by using the measured point set directly.

Other checks involve calculation of the tolerance zone and comparison of measured entities with the zone. The calculations required are dependent on the type of the entity and the zone.

As described in Chapter 7, geometrical tolerances can be used to constrain the relative positioning or orientation between geometric entities. In these cases, the geometric parameters of the measured datums are extracted and used to determine the rotational and translational transformations required to align the nominal datum entity in the model to the measured datum entity. These transformations are then applied to the parameters of toleranced nominal entities which reference the datum in order to obtain the expected parameters relative to the measured datums. By comparing the corresponding measured entity parameters against these expected entity parameters, the relative constraints can be checked.

The comparisons fall into three categories; tests on plane entities, tests on axis entities and tests on quadric entities. For details about various geometric tolerances, consult British Standards BS308 and Chapter 7 of this thesis.

8.4.2 Checking Tolerances Applied To Planes

The tests on planes are the most straightforward since they only involve projection of a set of points in a given direction, and distance calculations. All tolerance zones on planes are planar volumes. Further, since the points measured lie on the planes, the tests involve direct analysis of the point data by comparison with a nominally correct plane.

The FLATNESS test is the most straightforward. The tolerance restricts the variation of surface in the planar direction. A plane is fitted to the measured points using principal component analysis techniques (see Section 2.10) and the maximum distance between the measured points in a direction parallel to the planar normal is calculated. If this is within the given tolerance width, then the plane is within tolerance.

Tolerances of attitude applied to planes, such as SQUARENESS, PARALLELISM and ANGULARITY, reference datums, which may be axes or planes. The normal direction of the expected plane is calculated using the transformations described in Section 8.4.1 above. The maximum distance between the measured points in this direction is then calculated. As before, if this is within the given tolerance width, then the plane is within tolerance.

Tolerances of position again reference datums, and transformations are applied to determine the expected nominal planes. This time, since the location of the plane is being checked, the maximum distance between the measured points and the expected nominal plane in the direction of the planar normal is calculated. The expected nominal plane is assumed to be at the centre of the tolerance zone and so the calculated distances must lie within half the tolerance width. POSITION is handled this way. SYMMETRY references two datums which must be of the same entity type. Two nominal planes are calculated for this case, one for each datum, and a mean is taken as the expected plane. The test then proceeds as with the others.

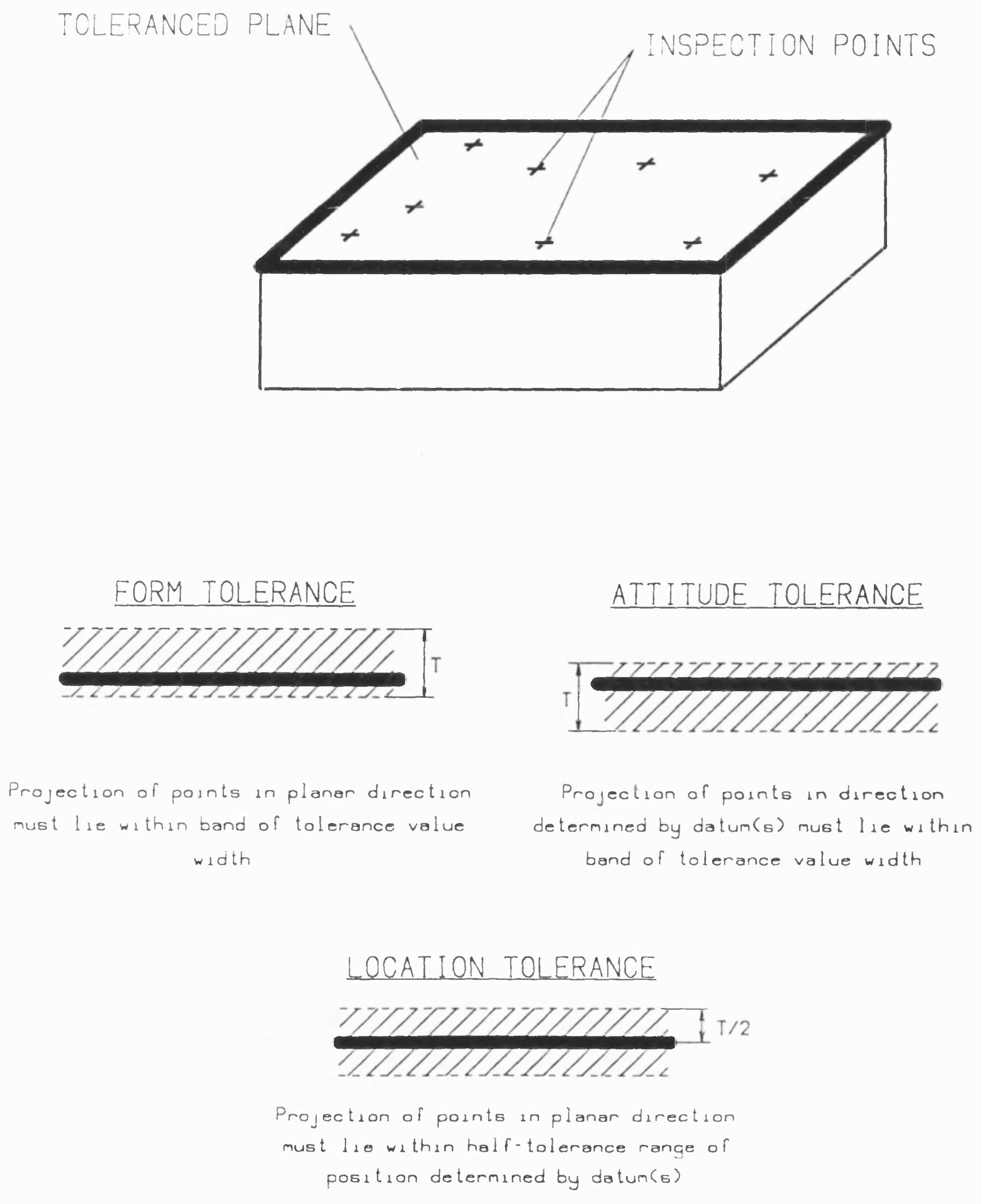

TOLERANCED PLANE
INSPECTION POINTS
FORM TOLERANCE
T
Projection of points in planar direction must lie within band of tolerance value width
ATTITUDE TOLERANCE
T
Projection of points in direction determined by datum(s) must lie within band of tolerance value width
LOCATION TOLERANCE
T/2
Projection of points in planar direction must lie within half-tolerance range of position determined by datum(s)

Figure 8.1 Interpretation of tolerances applied to planes.

8.4.3 Axis Tests

Axis tests involve comparison of lines with tolerance zones. The measured points themselves cannot be used directly to test the tolerance since the tolerance applies to an entity which cannot be measured directly, i.e the axis. Instead, geometric parameters which define the axis, namely the direction and location, have to be extracted from the measured points using the methods described in Section 8.2.3.

For tolerances which reference datums, the expected nominal test axis is derived from the datums using the appropriate transformations in the same way as described for the plane tests. For SYMMETRY tolerances, where there are two datums, the parameters of the test axis is calculated by taking the mean of the two sets of expected parameters which have been derived from the two datums.

The tolerance zone can be either planar area, planar volume or cylindrical volume. The orientation of the zone is obtained from the nominal (expected) axis, and if the zone type is planar area, the plane direction of the zone is specified within the tolerance attribute attached to the entity.

As described in Section 8.3.3, the set theoretic expression to which a tolerance applies defines a bounded nominal region. By replacing the nominal half spaces within this expression with the measured half spaces, the corresponding *measured region* is obtained, and this also will be bounded. The checks on axes involve calculating the range of the measured axis which lies within the measured region. This is done by ray-casting along the measured axis into the measured region and calculating the two intersection points. In order to check the tolerance, these points are compared against the tolerance zones as described below.

Tolerance of attitudes, such as SQUARENESS, PARALLELISM and ANGULARITY, require the orientation and width of the zone. The location tolerances such as POSITION, SYMMETRY and CONCENTRICITY in addition require the position of the zone. For such cases, the zone is centred about the nominal axis.

Fig 8.2(a) illustrates how attitude tolerance zones which are planar areas are checked. The test has to determine whether the segment of the axis which lies within the measured region also lies within the given tolerance zone. The end points of the measured axis within the region are calculated as described above. A test direction which is perpendicular to both the normal direction of the plane within which the zone lies and the nominal axis direction is calculated. The distance between the two points in this test direction is determined and if this distance is less than the tolerance width, then the axis lies within tolerance.

For location tolerance zones which are planar areas, the same calculation as for attitude planar areas is done to obtain the end points of the measured axis within the measured region and also to determine test direction. As described above, because the zone is for a location tolerance it is centred at the nominal axis, and so the distance between each of the measured axis end points and the nominal axis must be less than half the tolerance width if it is to lie within the specified tolerance.

Fig 8.2(b) illustrates the case for an attitude tolerance and a cylindrical volume tolerance zone. The two measured axis end points for the measured region are calculated as before. The distance between these points in a plane perpendicular to the nominal axis is calculated. For the measured axis to remain within tolerance, this distance must be less than the diameter of the cylindrical tolerance zone, and this diameter is exactly the tolerance width.

For location tolerance zones which are cylindrical volumes, the distance of each end point of the measured axis from the nominal axis is required. For each end point, the distance is calculated in a plane containing the nominal axis and the end point. The nominal axis is assumed to be at the centre of the tolerance zone and so each calculated distance must be less than the radius of the cylindrical zone, i.e. less than half the tolerance value.

Planar volume zones for attitude tolerances again require the measured axis end points within the measured region. The distance between these points is calculated in the direction of the normal to the nominal plane. This distance must be less than the tolerance value for the axis to lie within the specified tolerance.

Finally, planar volume zones for location tolerances require the distance between the measured axis end points and the nominal plane to be less than half the specified tolerance value (using the fact that the nominal plane is at the centre of the zone). The required calculations are made as described above.

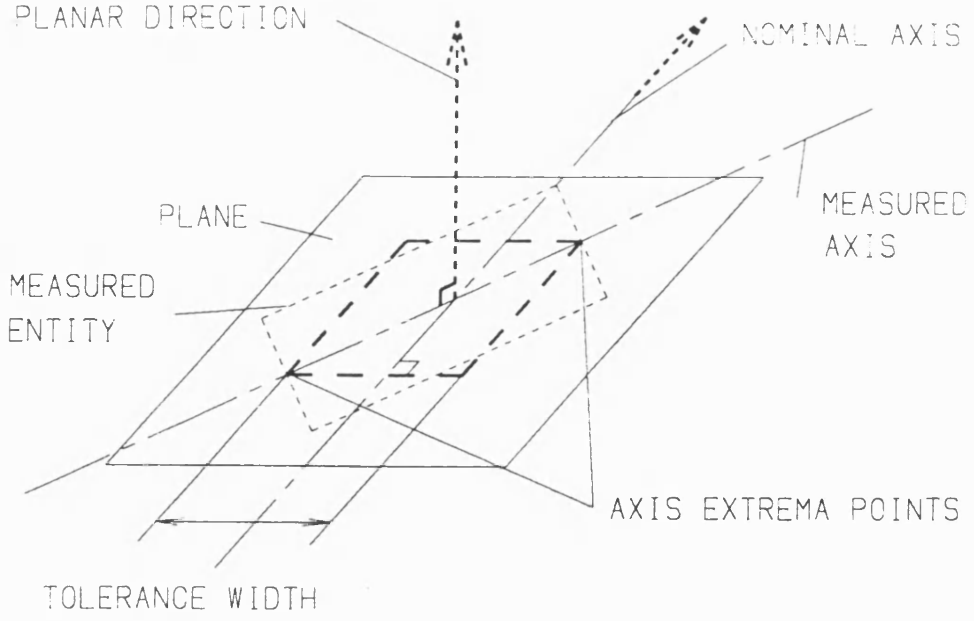


Figure 8.2(a) Interpretation of planar area tolerances applied to axes.

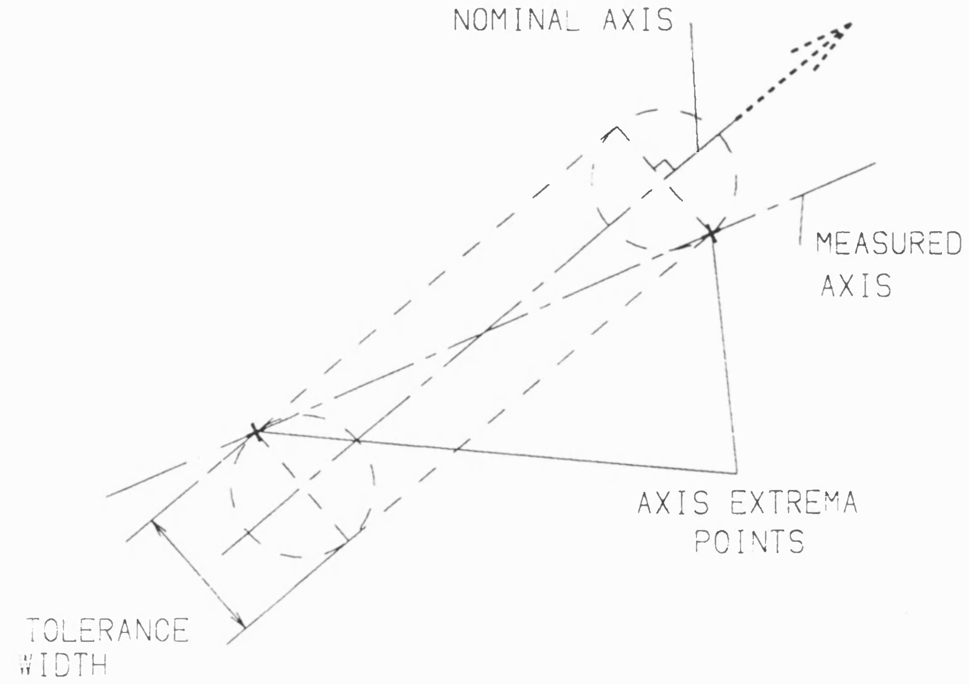


Figure 8.2(b) Interpretation of cylindrical volume tolerances applied to axes.

8.4.4 Sphere, Cylinder and Cone Tests

The tolerance of ROUNDNESS is assigned to the surface of cylinders, cones and spheres. Since the tolerances constrain the variation of the surface, the measured points can be used directly in testing the tolerance. However, in order to calculate the variations of the surface, the geometric parameters for the measured entity must first be extracted from the measured points using techniques described in Section 8.2.3.

When testing spheres (Fig 8.3(a)), the distance between the measured centre and each measured point is calculated. The maximum difference between these distances must be less than the tolerance width if the sphere is to satisfy the tolerance constraints.

When testing cylinders (Fig 8.3(a)), the distance between the measured axis and each measured point is calculated. The maximum difference between these distances must be less than the tolerance width if the cylinder is to satisfy the tolerance constraints.

When testing cones (Fig 8.3(b)), the half angle and location of apex of the measured cone is used to calculate the distances between the measured points and the measured axis. The maximum difference between these distances must be less than the tolerance width if the cone is to satisfy the tolerance constraints.

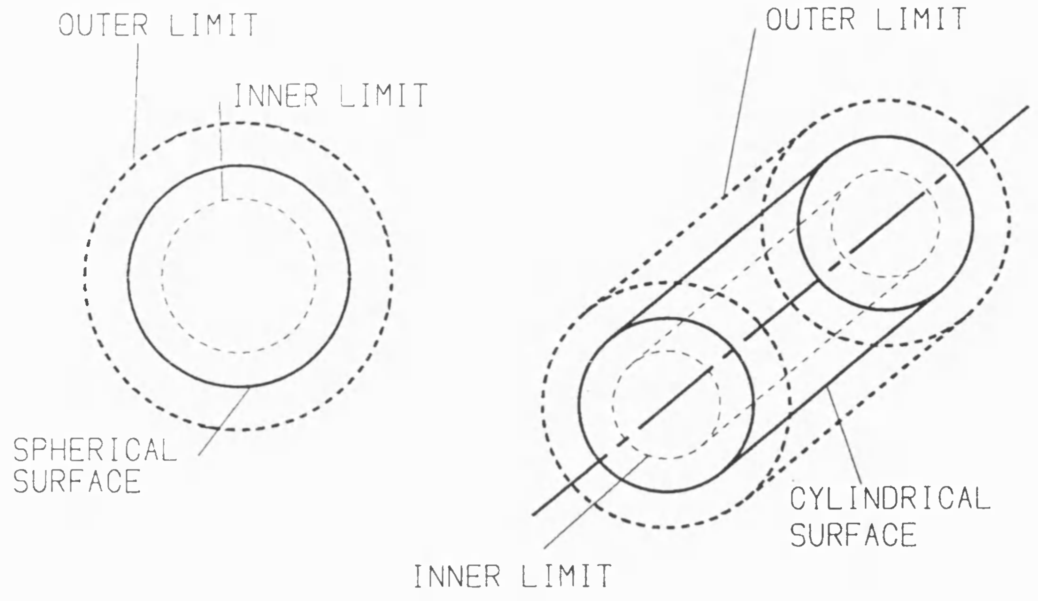


Figure 8.3(a) Tolerance zones for spheres and cylinders.

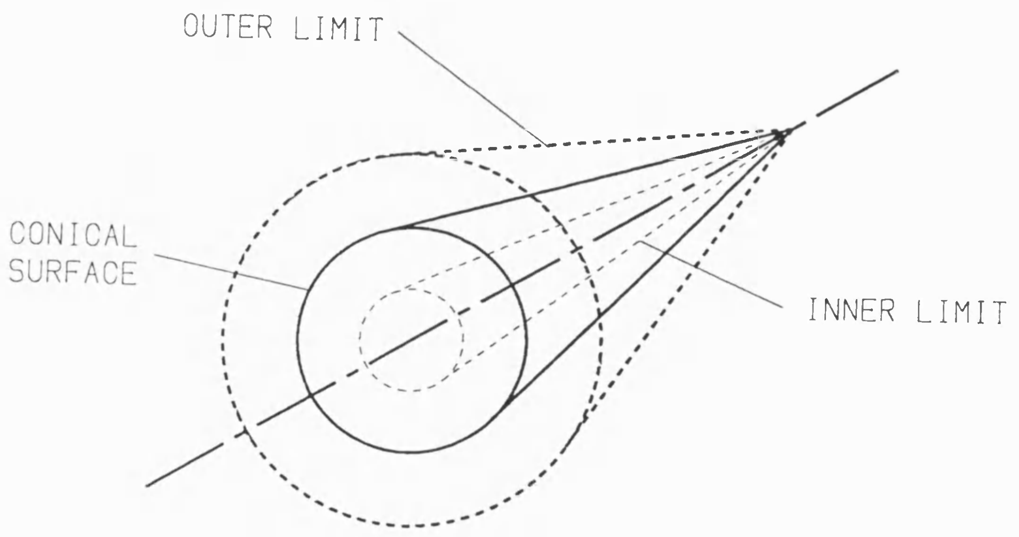


Figure 8.3(b) Tolerance zones for cones.

8.5 Conclusions

The methods described in this chapter have been implemented and used with success to analyse results of the component shown in Plate 6.1. Each of the toleranced features are considered in turn and the results are shown in Fig 8.4.

The set theoretic solid model is used throughout to determine the geometric parameters of nominal entities, which are then used for the comparisons. Specified datums also are used in determining these nominal test parameters and as a general rule should be inspected in order to allow constraints on relative positions and orientations to be checked.

The procedures return values which reflect the deviation of the measured features from the nominal features.

```
Tolerance Type : CONCENTRICITY  Zone Type: CYLINDRICAL VOLUME

Nominal Entity description
--------------------------
Entity type : AXIS   Axis type : CYL_AXIS

Location of Axis : 35.000000 70.000000 150.000000
Axis Direction   : 0.000000 0.000000 1.000000
Cylinder Radius  : 10.000000

Number of Measured Points · 13

Measured Entity
---------------

Location of Axis : 34.916999 71.055946 149.994059
Axis Direction   : -0.000770 0.005607 0.999984
Cylinder Radius  : 9.981178

Datum Description
-----------------

Datum Label : G_DTM   Datum Type : AXIS   Axis Type : CYL_AXIS

Location of Axis : 35.000000 70.000000 150.000000
Axis Direction   : 0.000000 0.000000 1.000000
Cylinder Radius  : 20.000000

Number of Measured Points : 14

Measured Datum
--------------

Location of Axis : 34.736282 70.122606 149.999487
Axis Direction   : -0.002504 -0.001404 0.999996
Cylinder Radius  : 19.435869

Expected Nominal Axis
---------------------

Location of Axis : 35.077143 70.228697 149.744737
Axis Direction   : -0.002504 -0.001404 0.999996
Cylinder Radius  : 20.000000

Axis / Measured Entity Intersection points
------------------------------------------

Point 1: 35.013122 70.033136 -5.158507
Point 2: 34.990037 70.201352 24.841013

Test Result
-----------

Test Value for Point 1 : 0.612209
Test Value for Point 2 : 0.448288

Tolerance Width : 0.500000
Result of Test (0=In Tolerance, -1=not): -1
```

Figure 8.4 Results of CYLINDRICITY tolerance checking for stepped hole in component shown in Plate 3.5.

CHAPTER 9

CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

9.1 Conclusions

The project described in this thesis has demonstrated successfully how set theoretic solid modelling can be used to automate the planning and execution stages of CMM inspection. The algorithms have been developed to work in conjunction with a set theoretic solid modelling system at Bath University although the principles are applicable to any set theoretic modelling system.

The ability to attach attributes to models, such as geometrical tolerances, has enabled the non-geometric requirements of the inspection process to be incorporated within the model description. Two types of attributes have been identified; those which are totally independent of the position and orientation of the entities to which they apply, such as colour; and those which are dependent on the position and orientation of the entities, such as geometric tolerances. The distinction is needed to determine whether or not a specified attribute remains valid after the entity to which it applies has been subjected to geometric transformations.

The use of attributes provides a means of control both over the distribution of inspection points and over the number of points generated. The final selection of inspection points depends on the analysis for which the measurements are to be used. For example, it is likely that different parameter estimation routines will benefit from different types of distributions. It has been demonstrated that generating an inspection point set by starting with an initial set of surface points and eliminating points from this set is a successful way of controlling the size and point distribution of the resulting point set.

In previous work, model growing techniques have been investigated in the context of obstacle configuration space (see Section 2.8.2) in path-planning. None of these techniques have used set theoretic solid models. This project has developed a method of approximating the configuration space of a component which corresponds to motion of a CMM probe by growing set theoretic models, and demonstrated how the resulting models can be used to automatically generate a collision-free inspection path for a CMM measuring probe.

Time-efficiency of the inspection path is controlled using a cost function. This enables the efficiency of the path to be related to a particular coordinate measuring machine; each machine can have its own cost function which will contain parameters to describe its performance characteristics.

Finally, the project has successfully demonstrated how the measured results can be used together with the toleranced set theoretic model to check that the component meets the required specification. This has been done for commonly used geometrical tolerances by using guidelines given in British Standards BS308.

The assignment of geometrical tolerances as defined in British Standards BS308 to two-dimensional drawings is incomplete in the sense that the range of the tolerance zone is not fully defined. The extra information required to bound the zone can be added to the model by only specifying tolerances to bounded entities within the component.

The algorithms have been used successfully on a range of models containing planar half spaces and Appendix 2 shows the results of using the system to inspect a component consisting of various geometric entities. (The described algorithms have been implemented on a Sun 3/160 Workstation).

9.2 Further Work

9.2.1 Growing

As described in Chapter 4, growing set theoretic models is not directly extendable to models containing curved surface half spaces. Further work may be done in this area, starting with the consideration of models which use a restricted class of curved surfaces, such as simple quadrics (i.e. spheres, cylinders and cones). Attributes can be attached to aid in identification of these surfaces, as described in Chapter 8, and appropriate growing procedures can be investigated for each geometric entity which would involve deriving a new polynomial from the original half space polynomial. Again care will be needed in maintaining the topology, and another area for further work is to investigate whether similar correction techniques as those in Chapter 4 can be used. As an alternative to this, techniques of automatically faceting the quadrics can be investigated to enable direct application of the techniques described in Chapter 4.

Investigations can be done to establish whether methods of tree reconstruction are more successful for curved surface modellers where increases in the number of half spaces will not be as large as they are for faceted models. Also, work can be done to investigate whether attributes can be used to convey useful information to the reconstruction algorithms in order to reduce the time taken and the size of the model.

9.2.2 Point Generation

As suggested in Chapter 5 the procedures and algorithms used to analyse the measurement results should be considered during the initial selection of inspection points. In the work described in this project, the analysis of measured points is based on least square fits to measured point sets. Given that the measured points are collected at the selected inspection points, further work is possible in determining a selection of points whose distribution is the "best" for the least-squares fitting algorithms.

9.2.3 Path Generation

Extending the path generation algorithms to accommodate swivel probes is clearly a topic for further work. Techniques from robot-path planning may be needed. One approach is to only rotate the probe when positioned at a safe distance from the component. In this way motion close to the component consists only of translations and hence can be analysed using techniques described in Chapter 6, with a different grown model for each orientation of the probe.

The path-planning algorithm in Chapter 6 generates an initial time-efficient test path and then tests this path against each of the grown models generated for the probe parts. The path is translated for each model so that it represents the motion of the appropriate probe part as the path is followed. When collisions occur, the path is modified and this will effect the efficiency of the original path. Further work can be done on developing alternative strategies to this. For example, investigation can be made to ascertain whether a better alternative is to translate the models rather than some initial test path. By unioning these translated models together the free-space available to all the probe parts is represented in a single model and, in principle, this model can be used to generate a collision-free path which does not require further modification.

Finally further work can be done in determining whether attributes attached to the model can be used to create more efficient paths for the probe.

9.2.4 Attributes

Chapter 8 describes how attributes are used to extract the geometric parameters of entities. Currently such parameter information attributes are applied "manually" to the various entities as the model is built. Such information has already been used to build the entity in the first place and a topic for further work is to investigate methods of deriving and assigning such attributes automatically when the component is built.

9.2.5 Analysis

Automatically extracting geometric parameter information from the solid model would be a useful tool for analysis and would remove the need of assigning such redundant information using attributes.

References

1. Gilheany, R and Treywin, E.T, "Developments in 3-D Measuring Machines & Associated Software," *Proc. Metrology Conference NELEX 80 Int.*, L K Tool & Co Ltd, 7 - 9 October. paper 2.2

2. Black, S.P, "Utilisation of 3-Coordinate Measuring Machines," *Proc. Metrology Conference NELEX 80 Int.*, Caterpillar Tractor Company Ltd, 7 - 9 October. paper 2.4

3. Bosch, J.A, "Flexible Inspection for Flexible Manufacturing," *Machine and Tool BLUE BOOK*, pp. 52 - 56, September 1988.

4. Raja, J and Sheth, U.P, "Integration of Inspection Into Automated Manufacturing System," *Recent Developments in Production Research*, pp. 119 - 124, Elsevier Science Publishers B.V., Amsterdam, 1988.

5. Park, H.D and Mitchell, O.R, "CAD Based Planning and Execution of Inspection," *Proc Computer Society Conference on Computer Vision & Pattern Recognition*, pp. 858 - 863, 1988.

6. Groover, M.P and Zimmers Jr. E.W, *CAD/CAM : Computer-Aided Design and Manufacturing,* Prentice-Hall, New Jersey, USA, 1984.

7. Atkins, N.W and Derby, S, "An Interactive Graphics Application for Computer Aided Development of Inspection Programs for Coordinate Measuring Machines," *Proc 15th Des Auto Conf in Computer Aided and Computational Design*, pp. 213-221, Montreal, Sept 17-21 1989.

8. Liddle, B, "Coordinate Measuring Machines Keeping Pace with NC Machine Tool Users," *Proc. 6th Int. Conf. on Automated Inspection and Product Control*, pp. 213 - 236, Ferranti Ltd U.K., Birmingham, 27-29 April 1982.

9. Groover, M.P, *Automation, Production Systems, and Computer Integrated Manufacturing*, Prentice-Hall, New Jersey, USA, 1987.

10. Bowman, I, "Is CMM Integrations Missing Link?," *Automation: Journal of Automated Control*, vol. 23 No 5, pp. 30 - 33, July 1987.

11. Quality Today, *Three Dimensional Coordinate Measuring Machine Survey*, pp. s17-s32, January 1989.

12. Sostar, A, "Coordinate Measuring Techniques in Quality Assurance," *Robotics and Computer-Integrated Manufacturing*, vol. 4 No 1/2, pp. 259 - 265, Great Britain, 1988.

13. Rquicha, A.A.G and Voelcker, H.B, "Solid Modelling: A Historical Summary and Contemporary Assessment," *IEEE Computer Graphics and Applications*, pp. 9 - 24, March 1982.

14. Requicha, A.A.G and Voelcker, H.H, "Solid Modelling: Current Status and Research Directions," *Production Automation Project*, Univ. of Rochester, October 1983.

15. Requicha, A.A.G, "Representations for Rigid Solids: Theory, Methods and Systems," *Computing Surveys*, vol. 12 No 4, pp. 437 - 461, December 1980.

16. Brown, C.M, "Some Mathematical and Representational Aspects of Solid Modelling," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.

PAMI-3 No 4, pp. 444 - 453, 4 July 1981.

17. Howard, K.L, "Solid Modelling: its Application in the Real World of Engineering and Industrial Design," *Computer Graphics '85*, pp. 365 - 380, Online Publications, Pinner, U.K, 1985.

18. Wilson, P.R, "Euler Formulas and Geometric Modelling," *IEEE Computer Graphics and Applications*, pp. 25 - 36, August 1985.

19. Woodwark, J.R, "Taming Set Theoretic Solid Models," *Proc. BCS Conf. State of The Art Seminar, New Tools for Shape Modelling*, London, Thurs 18th May 1989.

20. Woodwark, J.R and Quinlan, K.M, *The Derivation of Graphics from Volume Models by Recursive Subdivision of the Object Space.*

21. Woodwark, J.R and Quinlan, K.M, "Reducing the Effect of Complexity on Volume Model Evaluation," *Computer-Aided Design*, vol. 14 No 2, pp. 89 - 95, University of Bath, 1982.

22. Woodwark, J.R, "Elimination of Redundant Primitives from Set Theoretic Solid Models Using Constituents," *U.K.S.C Report*, April 1988.

23. Woodwark, J.R, "Generalizing Active Zones for Set Theoretic Solid Models," *Internal Report, U.K.S.C Report 182, IBM*, Jan 1988.

24. Zhang, D and Bowyer, A, "CSG Set Theoretic Solid Modelling and NC Machining of Blend Surfaces," *Proc. 2nd ACM Symposium on Computational Geometry*, New York, June 1986.

25. Woodwark, J.R, *Blends in Geometric Modelling,* IBM (UK) Scientific Centre, Winchester, Winchester.

26. Middleditch, A.E and Sears, K.H, *Blend Surfaces for Set Theoretic Volume Modelling Systems,* Poly of Central London, Tech Memo 84:4, 1985.

27. Warren, J, "Blending Algebraic Surfaces," *ACM Trans Graphics*, vol. 8 No 4, pp. 263 - 278, Oct 1989.

28. Bowyer, A, Davenport, J.H, Milne, P.S, Padget, J, and Wallis, A, "Applications of Computer Algebra in Solid Modelling," *Proc 1987 Eurocal Conference, Liepzig.*

29. Woodwark, J.R and Bowyer, A, "Better & Faster Pictures From Solid Models," *Computer Aided Engineering*, pp. 17 - 24, February 1986.

30. Wallis, A.F and Woodwark, J.R, "Interrogating Solid Models," *Proc. CAD Conf.*, 1984.

31. Woodwark, J.R, "Shape Models In Computer Integrated Manufacture - A Review," *Computer-Aided Engineering*, pp. 103 - 112, IBM United Kingdom Scientific Centre, June 1988.

32. Wallis, A, *Toolpath Verification Using C. S. G.,* PhD Thesis, 1989.

33. Martin, R.R and Stephenson, P.C, "Sweeping of Three-Dimensional Objects," *Computer-Aided Design*, vol. 22 No 4, pp. 223 - 234, May 1990.

34. Requicha, A.A.G and Tilove, R.B, "A Null Object Detection Algorithm for Constructive Solid Geometry," *Communications of the ACM*, vol. 27 No 7, pp. 684 - 694, July 1984.

35. Woodwark, J.R, "Some Speculations on Feature Recognition," *Computer-Aided Design*, vol. 20 No 4, pp. 189-196, May 1988.

36. Jared, G, "The Feature Recognition Battle - Latest From The Front," *Proc. BCS Conf. State of The Art Seminar, New Tools for Shape Modelling*, London, Thurs 18th May 1989.

37. Perng, D.B, Chen, Z, and Li, R, "Automatic 3D Machining Feature Extraction from 3D CSG Solid Input," *Computer Aided Design*, vol. 22 No 5, June 1990.

38. Sabella, P and Carlbom, I, "An Object-Oriented Approach To The Solid Modeling of Empirical Data," *IEEE Computer Graphics & Applications*, pp. 24 - 35, September 1989.

39. Alagar, V.S, Bui, T.D, and Periyasamy, K, "Semantic CSG Trees for Finite Element Analysis," *Computer-Aided Design*, vol. 22 No 4, pp. 194 - 198, May 1990.

40. Tan, S.T and Yuen, M.M.F, "Integrating Solid Modelling with Finite-Element Analysis," *Computer-Aided Engineering Journal*, pp. 133 - 137, August 1986.

41. Moore, R E, *Methods and Applications of Interval Analysis,* S.I.A.M Philadelphia, 1979.

42. Pai, S.N, "Interval Analysis," *MSc Thesis, University of Bath*, 1972.

43. Hansen, E.R, *A Generalized Interval Arithmetic*, pp. 7 - 18.

44. Goldfeather, J and Molnar, S, "Near Real-Time CSG Rendering Using Tree Normalisation and Geometric Pruning," *IEEE Computer Graphics and Applications*, pp. 20 - 28, 1989.

45. Lee, Y.C and Jes, K.F.J, "A New CSG Tree Reconstruction Algorithm for Feature Representation," *Proc. Computer in Engineering 1988*, pp. 521 - 528, July 31 - Aug 4 1988.

46. Lee, Y.C and Fu, K.S , "Machine Understanding of CSG Extraction and Unification of Manufacturing Features," *IEEE Computer Graphics and Applications*, vol. 7 No 1, pp. 20 - 32, Jan 1987.

47. British Standards Institute, "British Standard Guide to the Assessment of Position, Size, and Departure from Nominal Form of Geometric Features.," *British Standards*, vol. BS 7172, London, 1989.

48. Little, J.D.C, Murty, K.G, Sweeney, D.W, and Karel, C, "An Algorithm for the Travelling Salesman Problem," *Operations Research*, vol. 11, p. 972, 1963.

49. Lin, S, "Computer Solutions of Travelling Salesman Problem," *System Tech.*, vol. 44, pp. 2245-2269, 1965.

50. Aarts, E.H.L, Korst, J.H.M, and Van Laarhoven, P.J.M, "A Quantitative Analysis of the Simulated Annealing Algorithm: A Case Study for the Travelling Salesman Problem," *Journal of Statistical Physics*, vol. 50 No 1/2, pp. 187 - 206, 1988.

51. Vaidya, P.M, "An O(n log n) Algorithm for the All-Nearest-Neigbors Problem," *Discrete Computational Geometry*, vol. 4, pp. 101 - 115, 1989.

52. Alagar, V. S, Bui, T.D, and Periyasamy, K, "Reasoning System for Solid Modelling Techniques Applicable to Robotics," *Proc IFAC Robot Control*

(SYROCO '88), pp. 493 - 498, Karlsruhe, Oct 5-7 1988.

53. Cameron, S, "Efficient Intersection Tests for Objects Defined Constructively," *International Journal of Robotics Research*, vol. 8 No 1, pp. 3 - 25, February 1989.

54. Cameron, S, "Modelling Space and Time," *New Tools for Shape Modelling, BCS Conference Documentation Displays Group*, London, May 1989.

55. Mirolo, C and Pagello, E, "A Solid Modelling System for Robot Action Planning," *IEEE Computer Graphics and Applications*, pp. 55 - 69, January 1989.

56. Kondo, K and Kimura, F, "Collision Avoidance using a Free-space Enumeration Method Based on Grid Expansion," *Advanced Robotics*, vol. 3 No 3, pp. 159 - 175, 1989.

57. Lozano-Perez, T and Wesley, M. A, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Communications of the ACM*, vol. 2 No 10, pp. 560 - 570, 1979.

58. Lozano-Perez, T, "Spatial Planning: A Configuration Space Approach," *IEEE Transactions on Computers*, vol. C - 32(2), pp. 108 - 120, Feb 1983.

59. Fu, L-C and Liu, D-Y, "An Efficient Algorithm for Finding a Collision-free Path Among Polyhedral Obstacles," *Journal of Robotic Systems*, vol. 7 No 1, pp. 129 - 137, 1990.

60. Shah, J. J. and Miller, D. W, "A Structure for Supporting Geometric Tolerances in Product Definition Systems for CIM," *Manufacturing Review*, vol. 3 No 1, March 1990.

61. Requicha, A.A.G and Chan, S.C, "Representation of Geometric Features, Tolerances, and Attributes in Solid Modelers Based on Constructive Geometry," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 3, September 1986.

62. Roy, U and Liu, C.R, "Feature-Based Representational Scheme of a Solid Modeler for Providing Dimensioning and Tolerancing Information," *Robotics and Computer-Integrated Manufacturing*, vol. 4 No 3/4, pp. 335-345, 1988.

63. Ranyak, P S and Fridshal, R, "Features for Tolerancing a Solid Model," *Computers in Engineering ASME Conf Proc*, vol. 1, pp. 275-280, 1988.

64. Bernstein, N S and Preiss, K, "Representation of Tolerance Information in Solid Models," *Proc 15th Design Automation Conference, ASME Des Eng Div*, vol. 19 No 1, pp. 37 - 48, Montreal, 1989.

65. British Standards Institute, "Engineering Drawing Practice Part 3 : Geometrical Tolerancing," *British Standards*, vol. BS 308, London, 1972.

66. Fleming, A, "Geometric Relationships Between Toleranced Features," *Artificial Intelligence*, vol. 37 , pp. 403-412, 1988.

67. Etesami, F, "Tolerance Verification through Maunufactured Part Modelling," *Proc 15th Design Automation Conference - ASME Des Eng Div*, vol. 19 No 1, pp. 37-48, Montreal, 1989.

68. Jayaraman, R and Srinivasan, V, "Geometric Tolerancing I & II," *IBM J Res Develop*, vol. 33 No 2, pp. 90 - 124, March 1989.

69. Efimov, N, "An Elementary Course In Analytical Geometry (Parts I & II)," *The Commonwealth and International Utility of Science, Technology, Engineering and Liberal Studies.*

70. Cemal, C, "Component Inspection," *Phd Thesis University of Bath* , 1989.

71. Pratt, V, "Direct Least-Squares Fitting of Algebraic Surfaces," *Computer Graphics*, vol. 21 No 4, pp. 145 - 152, July 1987.

72. Forbes, A.B, "Least Squares Best-Fit Geometric Elements," *Internal Report, NPL Report DITC 140/89*, April 1989.

73. Elmaraghy, W. H., Elmaraghy, H. A., and Wu, Z, "Determination of Actual Geometric Deviations Using Coordinate Measuring Machine Data," *Manufacturing Review*, vol. 3 No 1, pp. 32 - 39, March 1990.

APPENDIX 1

COLLECTING THE MEASUREMENTS

A1.1 Introduction

Having generated the probe path, the component has to be measured using the CMM. This chapter describes how measurements were collected using a PC-controlled CMM.

A1.2 The Hardware

The CMM consists of three moving axes which are driven by stepper motors. Each axis can be driven at one of seven different speeds. The speeds are selected as integers by the PC and sent within a coded string to the CMM. Thus speed seven is, for example, much faster than speed one. The axes have high inertia and air bearings are used to enable smoother motion.

If changes in speed are made too sharply from one direction to another, the motors will slip and the axes will come to a rest. Similarly, if the probe hits a surface too fast, the CMM hardware will bring motion to a stop. Thus motion close to the component needs to be reasonably slow.

The PC is a standard IBM compatible machine fitted with a digital I/O card for sending signals to and receiving signals from the CMM.

A1.3 Controlling the Probe

Since only speeds and not positions are controlled, and the real values of these speeds is dependent on external conditions of the environment, driving the probe in an arbitrary direction at slow speeds was difficult. For this reason, the direction for probing points during measurements was restricted to axial directions, and the axis direction was selected as that closest to the surface normal at the surface inspection point. This surface information is available within the solid model.

For driving the probe between points which are far apart, the axis speed is calculated from the distance between the two points in the axial direction. The probe is driven at the fastest speed defined by the distance until close to its destination and is then brought gradually to a stop.

Such control makes it difficult to drive the probe along an arbitrary straight line and this causes problems when motion is close to the component, where deviation from the expected line of motion sometimes leads to unexpected collisions.

The probe path consists of two types of steps. The first specifies accurate positioning of the probe and precedes the taking of a measurement. The second is used when moving the probe between non-measure points, like a safe-plane or obstacle avoidance step described in Chapter 6, when accurate positioning is not as important. Two different position tolerances were used to account for these.

The control was implemented in the C language on the PC.

A1.4 Probing the Points

A touch-sensitive probe is used to collect the measurements. Measurements are taken by moving the probe slowly towards the object surface until contact is made.

Before the probe hits the surface it is in a state known as *armed.* Once it hits the surface, however, the probe becomes disarmed and the CMM hardware takes control by moving the probe back along the direction it was travelling. This is known as *backing-off* and is controlled entirely by the CMM hardware. A measurement is taken by the PC when the probe first re-arms and this required continuous monitoring of the state of the probe.

The measurement recorded is for the centre of the probe. The correct coordinate for the surface points needed to be calculated and this is done using the probe radius and surface normal obtained from the solid model.

The component is measured at the specified measuring points along the path and the results written to a file for further processing.

A1.5 Measuring the Component

The coordinate systems between the model and the CMM are unlikely to have the same origin. Further, the component need not be correctly orientated and hence a transformation matrix is required to align the two systems. A further complication arises with the CMM used to test the software since the coordinate system of the CMM is left-handed. This is accounted for by reflecting the path data in the x-axis.

The CMM has a measuring table whose coordinate in the z-direction is fixed. There is also a datum block on the CMM which is used to initialise parameters, such as the coordinates of the ends of the axes and the z-coordinate of the measuring table.

Attributes in the solid model are used to define three measure datum planes and these are probed before measuring begins. If the orientation of the component is approximately correct, the probing of these planes can be done automatically. An assumption is made that one of the measure datums is the surface of the component in contact with the measuring table. In this way, only two points on the second plane and a single point on the third need to be probed. These three points, together with the equation of the measuring table plane and the three measure datum planes are used to derive the transformation matrix.

APPENDIX 2

EXAMPLE OF ALGORITHMS

A2.1 Introduction

To illustrate the various aspects of the inspection planning system, the algorithms have been applied to a test shape consisting of planar faces, vertical and angled cylinders, and a triangular pocket [fig A2.1]. The probe model consists of two vertical cylinders and spherical probe tip. This chapter contains the results of applying the algorithms described in this thesis and highlights the properties of the system.

A2.2 Selecting Inspection Points

Plate A2.1 shows the points which are generated using the grid nodes as surface points. Attributes have been used to select various entities, including the angled face and cylinder, the pocket and the protruding block. The points are clearly regularly spaced on the planar faces, each with a different grid spacing. However, clusters of points are seen on the cylindrical entities.

Some points are also clearly unreachable, such as those in the bottom corners of the triangular pocket or those lying on the ledge of the stepped semi-cylindrical hole.

Fig A2.1 shows the points for the semi-cylindrical hole in more detail. A lower grid spacing assigned to the "ledge" has yielded more points in this example and Fig A2.2 shows the results after elimination of the unreachable points. The only surviving ledge points are those nearest the edge.

Points lying on the lower half of the semi-cylindrical hole have been classified as unreachable because of interference between the protruding block of the test rig and the probe head. Similarly, the points within the angled cylinder cannot be reached because of the vertical probe arm. Finally, points within the thin blind hole are also classified as unreachable because the probe sphere is unable to be positioned at the required offset point. Currently, the offset distance is fixed for all entities, but this may be assigned as a entity attribute and thus would allow smaller or larger offset distances.

Fig A2.3 shows the final selection of inspection points for this entity. Clusters have now been removed from the upper cylinder and the recommended number of points, 13, has been achieved.

A2.3 Growing for The Probe

Plate A2.2 shows the result of growing the model for the cylindrical probe arm. The crosses mark where new vertices have arisen and hence where topology changes have occurred. Vertices have appeared where the cylinders on the corners of the pocket which have shrunk to introduce extra solid at the union boundary with the pocket sides. New vertices also occur when the bounding plane of the lower semi-cylindrical entity shifts up into the model.

Plate A2.3 illustrates the corrected model. The semi-cylindrical hole has been reset to the full length of the block. However, vertices which occur at the boundary of cylinders and other planes have not been corrected. This is because the half-spaces which have introduced the new vertices also lie on vertices in the ungrown model and hence there are no shiftable half-spaces available to correct the vertices.

Fig A2.4, A2.5 and A2.6 illustrate the three grown models corresponding to the three probe segments. In the model grown for the probe tip, the blind hole is very thin and hence requires a very small offset distance for probing points. On the model for the probe arm, the angled cylinder has closed signifying that no points can be reached using the given probe. Finally, the model for the probe head illustrates how the approximation to the grown model through shifting half-spaces becomes more conservative when large growing displacements are considered. The cylindrical edges of the protruding block disappear as the middle segment of the block grows.

A2.4 Path Generation

The result of weighting the path so that motion in the x-axis direction is preferred over motion in the y-axis is shown in figures A2.7, A2.8 and A2.9 for various stages of a measuring path. The path is generated for three entities; the triangular pocket, the semi-cylindrical stepped hole and the angled face. (The y-axis direction is towards the top of the page). The three-dimensional nature of the path makes it difficult to visualise, however certain information can be glimpsed by comparing with the figures in A2.10, A2.11 and A2.12 where all axes are given equal weight.

The path starts with the angled face. For the even-weighted path, the nearest point is chosen in each case and the path moves down the face. Having missed a point on one end of the face, the path does a back-trace when it reaches the other end. This illustrates the entity-based approach of the path, where all points on the entity are considered before moving on, and the less-than optimal nature of nearest-neighbour path search. For the y-axis weighted path, the angled face is considered first again, but moving down the face is now more expensive than moving across. Thus the path does not miss a point at one end and hence does not have to back trace.

The evenly weighted path then selects the nearest entity, which is the pocket and works its way around there before finishing with the stepped hole. Similarly, the y-axis weighted path goes on to probe its cheapest neighbour, the stepped hole, and then goes on to inspect the pocket.

A2.5 Analysis

The results of inspection are shown in Fig A2.13. Most entities were found not to lie within the required tolerance. The slanted face is the worst as can be seen by considering the results in figures A2.13(b) and A2.13(d).

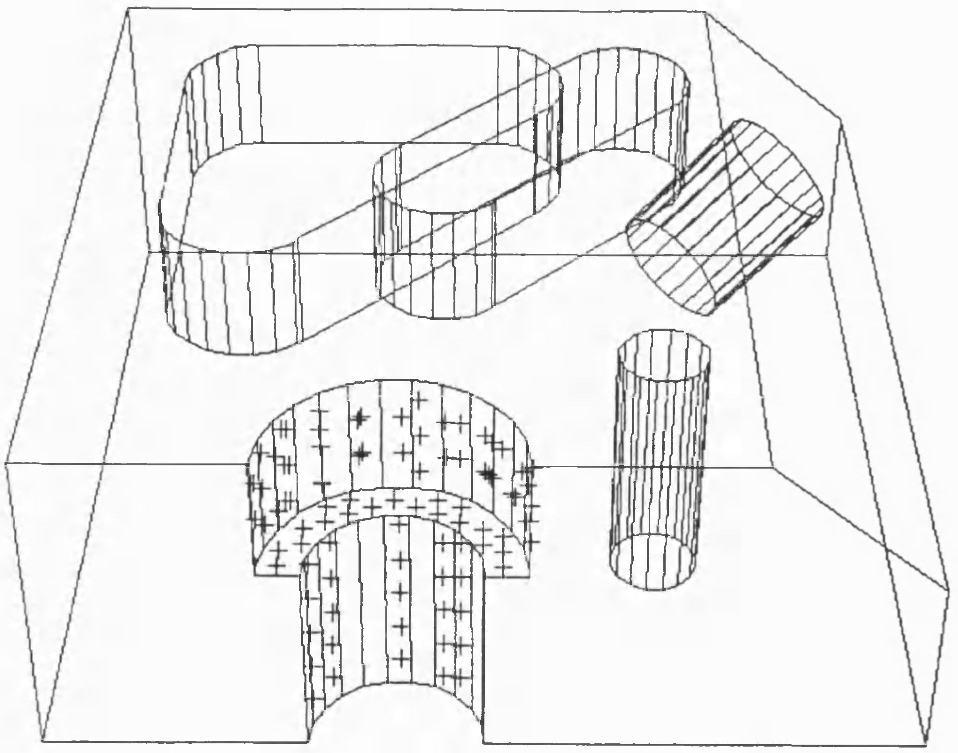


Figure A2.1 Surface points generated for stepped semi-circle feature of test rig.

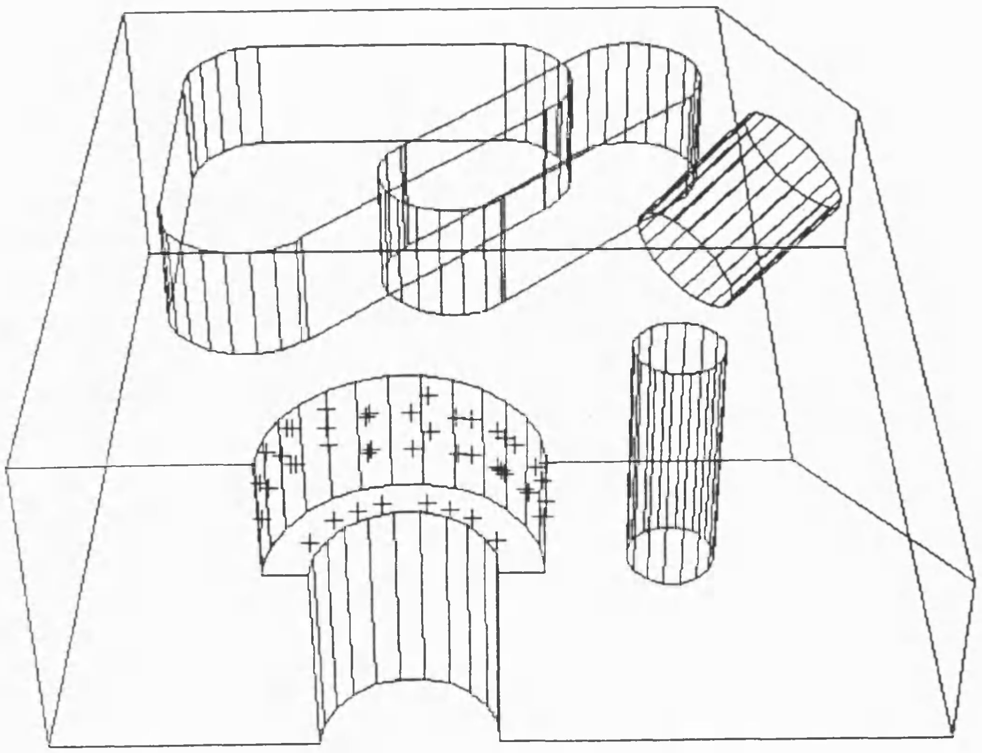


Figure A2.2 Reachable sub-set of points.

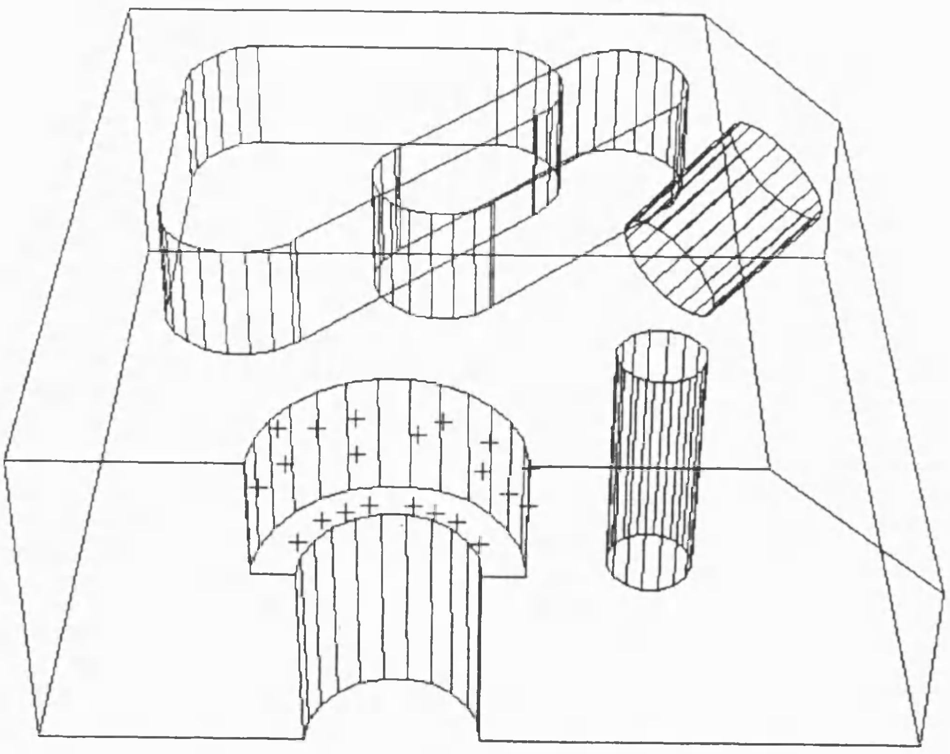


Figure A2.3 Final selected inspection points.

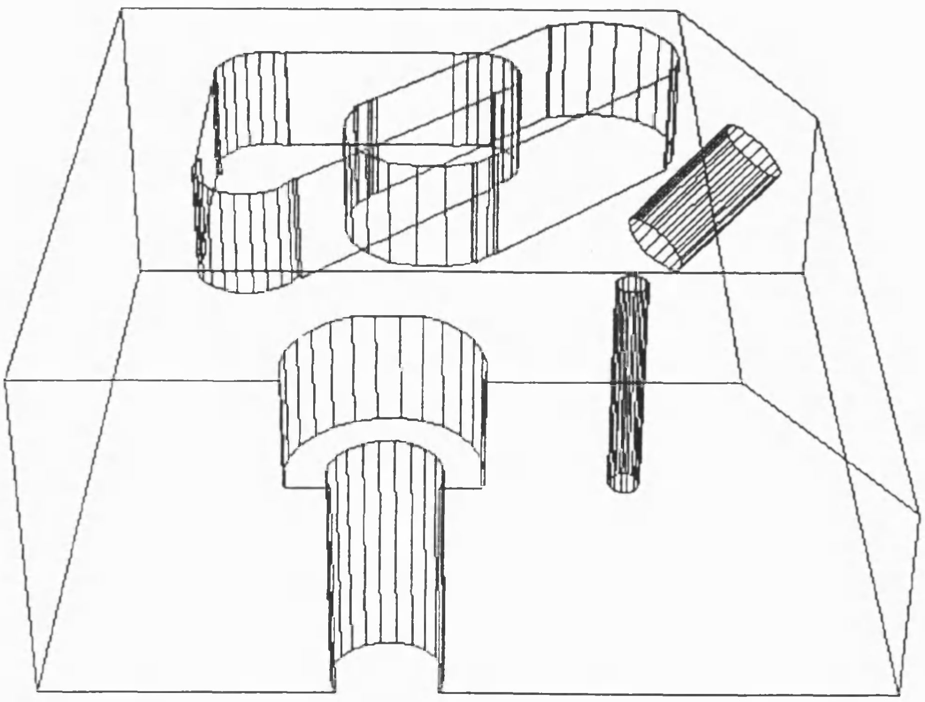


Figure A2.4 Test rig grown for probe tip.

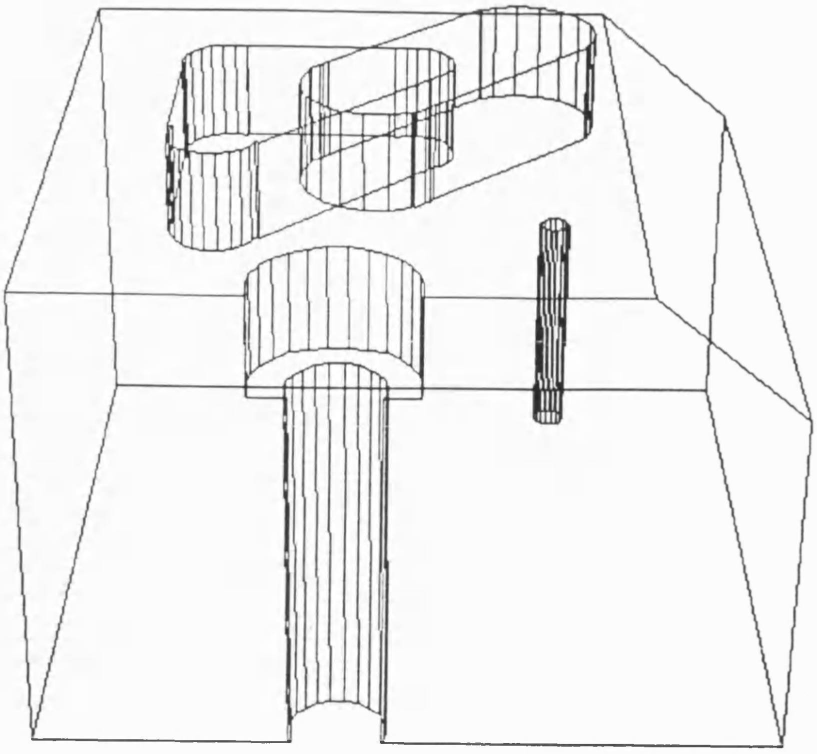


Figure A2.5 Test rig grown for probe arm.

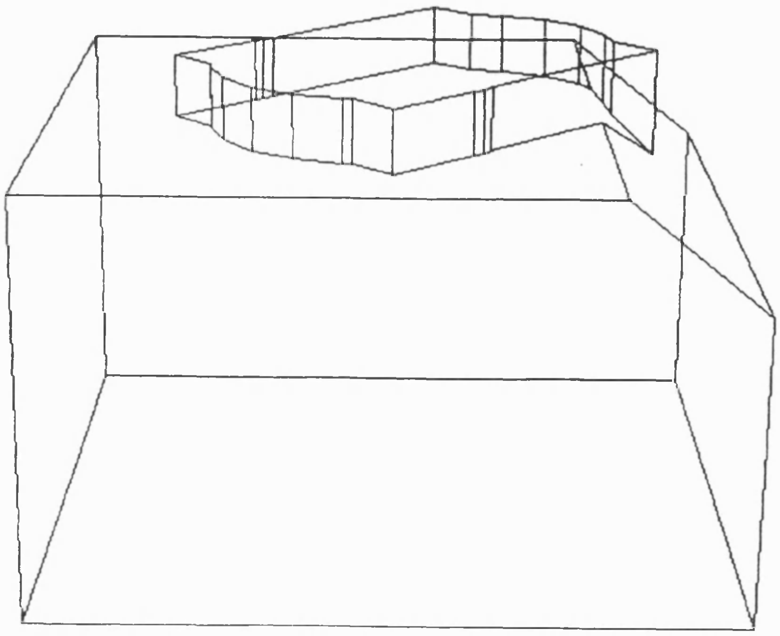


Figure A2.6 Test rig grown for probe head.

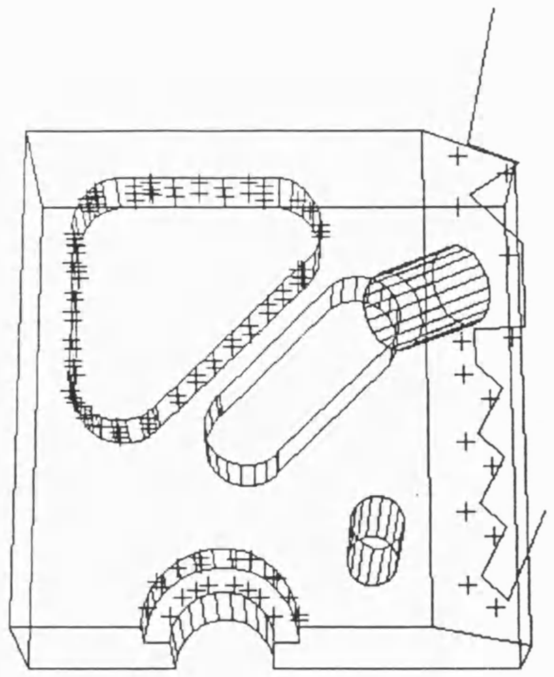


Figure A2.7 Inspection path for angled face feature of test rig with motion weighted against y-axis.

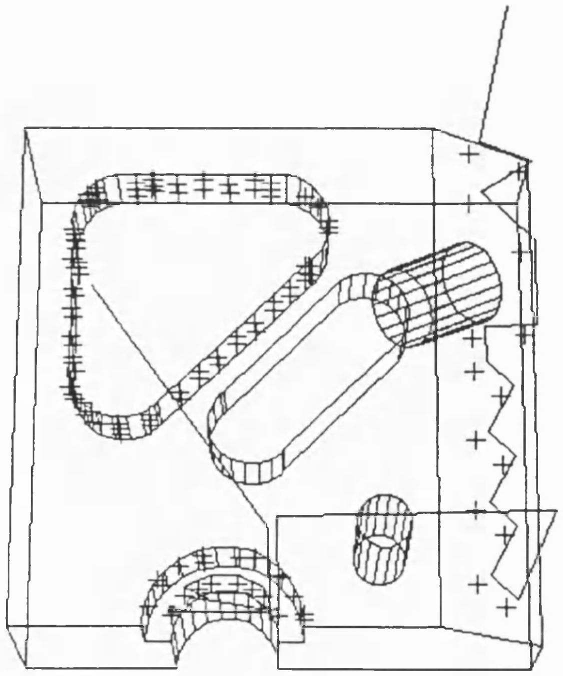


Figure A2.8 Inspection path for semi-circle feature of test rig with motion weighted against y-axis.

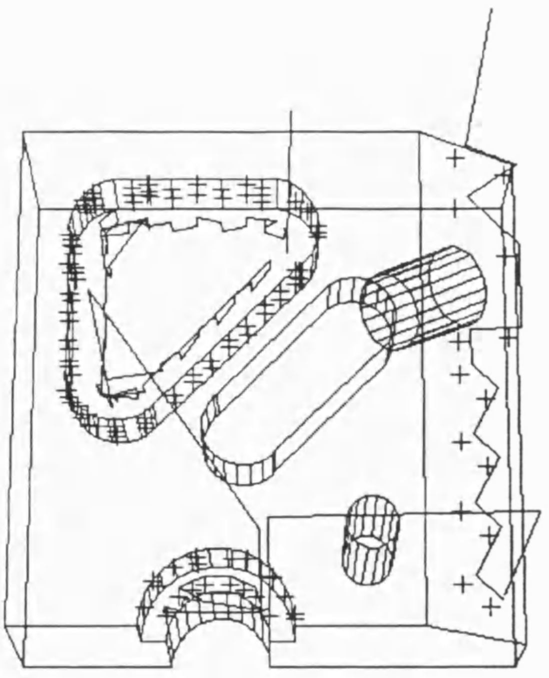


Figure A2.9 Inspection path for pocket feature of test rig with motion weighted against y-axis.

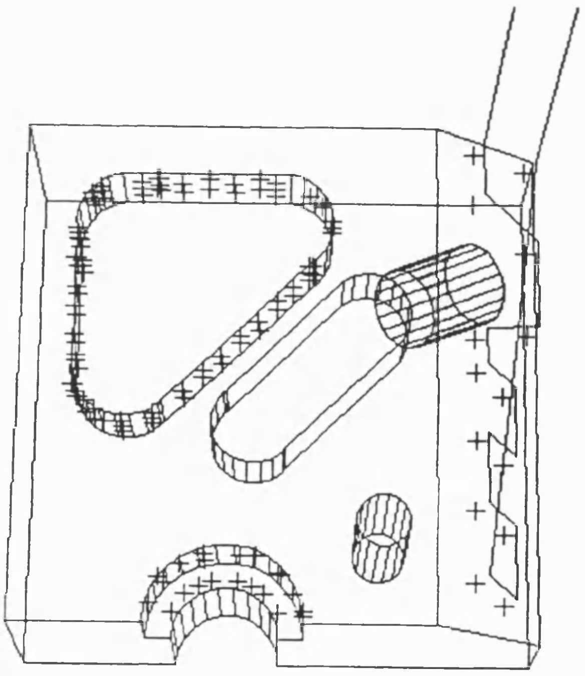


Figure A2.10 Inspection path for angled face feature of test rig with equal axis weighting.

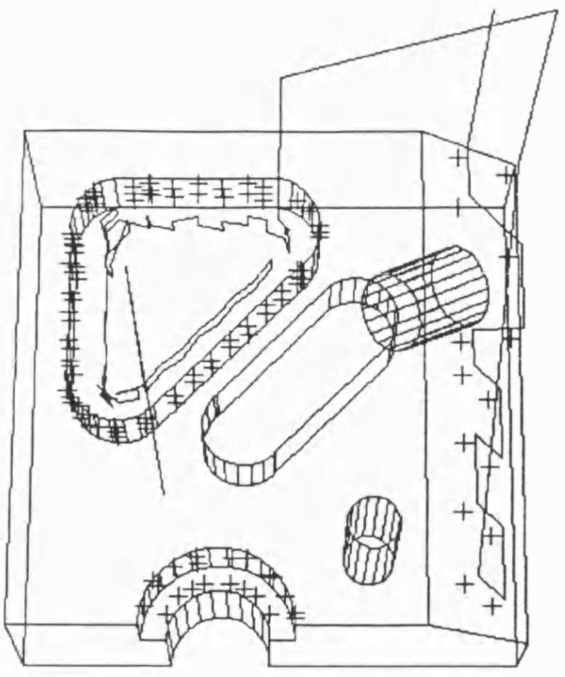


Figure A2.11 Inspection path for pocket feature of test rig with equal axis weighting.

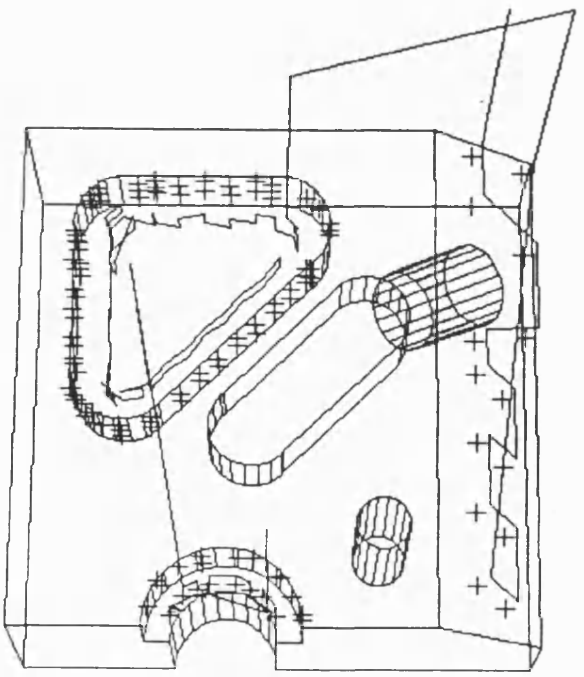


Figure A2.12 Inspection path for semi-circle feature of test rig with equal axis weighting.

```
Tolerance Type : SQUARENESS      Zone Type: PLANAR VOLUME

Nominal Entity Description
----------------------------

Entity Type : PLANE
Half Space Equation : 0.000000 -1.000000 0.000000 90.000000

Number of Measured Points   10

Measured Entity
----------------

Half Space Equation : -0.002750 -0.997208 -0.014665 90.604141

Datum Description
-----------------

Datum Label : X0_DTM   Entity Type : PLANE

Half Space Equation : -1.000000 0.000000 0.000000 0.000000

Number of Measured Points   0

Expected Nominal Entity
-----------------------

Test plane : 0.000000 -1.000000 0.000000 90.000000

Tolerance Test Result
---------------------

Tolerance Width : 0.100000
Range of Point Distances : 0.153110
Result of Test (0=in tolerance, -1=not): -1
```

Figure A2.13(a) Results of SQUARENESS tolerance test for test rig.

```
Tolerance Type : ANGULARITY      Zone Type: PLANAR VOLUME

Nominal Entity Description
---------------------------

Entity Type : PLANE
Half Space Equation : 0.644488 0.000000 0.764614 -82.143630

Number of Measured Points   14

Measured Entity
----------------

Half Space Equation : 0.597107 0.041647 0.798589 -79.774050

Datum Description
-----------------

Datum Label : Z1_DTM   Datum Type : PLANE

Half Space Equation : 0.000000 0.000000 1.000000 -40.000000

Number of Measured Points   0

Expected Nominal Entity
-----------------------

Half Space Equation :0.644438 0.000000 0.764614 -82.143630

Tolerance Test Result
---------------------

Tolerance Width : 0.500000
Range of Point Distances : 2.259281

Result of Test (0=in tolerance, -1=not): -1
```

Figure A2.13(b) Results of ANGULARITY tolerance test for test rig.

```
Tolerance Type : SQUARENESS     Zone Type: PLANAR VOLUME

Nominal Entity Description
----------------------------

Entity Type : PLANE
Half Space Equation : 0.000000 0.000000 1.000000 -25.000000

Number of Measured Points : 8

Measured Entity
----------------

Half Space Equation : 0.000487 -0.000668 0.999751 -23.088208

Datum Label : ZHOLE_DTM   Entity Type : AXIS  Axis Type : CYL_AXIS

Location of Axis : 40.000000 0.000000 25.000000
Axis Direction   : 0.000000 0.000000 1.000000
Cylinder Radius : 10.000000

Number of Measured Points : 0


Expected Nominal Entity
-----------------------

Half Space Equation : 0.000000 0.000000 1.000000 -25.000000


Tolerance Test Result
---------------------

Tolerance Width : 0.100000
Range of Point Distances : 0.012000
Result of Test (0=in tolerance, -1=not): 0
```

Figure A2.13(c) Results of SQUARENESS tolerance test for the test rig.

```
Tolerance Type : FLATNESS        Zone Type: PLANAR VOLUME

Nominal Entity Description
--------------------------
Entity Type : PLANE
Half Space Equation : 0.644488 0.000000 0.764614 -82.143630

Number of Measured Points : 14

Measured Entity
----------------

Half Space Equation : 0.597107 0.041647 0.798589 -79.774050

Expected Nominal Entity
-----------------------

Half Space Equation : 0.597107 0.041647 0.798589 -79.774050


Tolerance Test Result
---------------------

Tolerance Width : 0.100000
Range of Point Distances : 5.173707

Result of Test (0=in tolerance, -1=not): -1
```

Figure A2.13(d) Results of FLATNESS tolerance test for the test rig.

```
Tolerance Type : ROUNDNESS      Zone Type: CYLINDRICAL VOLUME

Nominal Entity Description
---------------------------

Entity Type : CYLINDER   Axis Type : CYL_AXIS

Location of Axis : 50.000000 80.000000 25.000000
Axis Direction : 0.000000 0.000000 1.000000
Cylinder Radius 10.000000

Number of Measured Points . 11

Measured Entity
----------------

Location of Axis : 49.576116 79.989677 25.009381
 Axis Direction : 0.022313 -0.017641 0.999595
 Cylinder Radius 10.466326

 Range of Point Distances : [10.527090,10.377974]
 Tolerance Width : 0.100000
 Result of Test (0=in tolerance, -1=not): -1
```

Figure A2.13(e) Results of ANGULARITY tolerance test for the test rig.

```
Tolerance Type : ANGULARITY      Zone Type: PLANE VOLUME

Nominal Entity Description
---------------------------

Entity Type : PLANE
Half Space Equation : -0.707107 0.707107 0.000000 -11.213210

Number of Measured Points : 16
Measured Entity
----------------
Half Space Equation : -0.705479 0.706960 0.005545 -11.396518

Datum Description
-----------------

Datum Label : X0_DTM   Entity Type : PLANE

Half Space Equation : -1.000000 0.000000 0.000000 0.000000

Number of Measured Points : 0

Expected Nominal Entity
-----------------------
Half Space Equation : -0.707107 0.707107 0.000000 -11.213210

Tolerance Test Result
---------------------

Tolerance Width : 0.100000
Range of Point Distances : 0.158552
Result of Test (0=in tolerance, -1=not): -1
```

Figure A2.13(f) Results of ROUNDNESS tolerance test for the test rig.

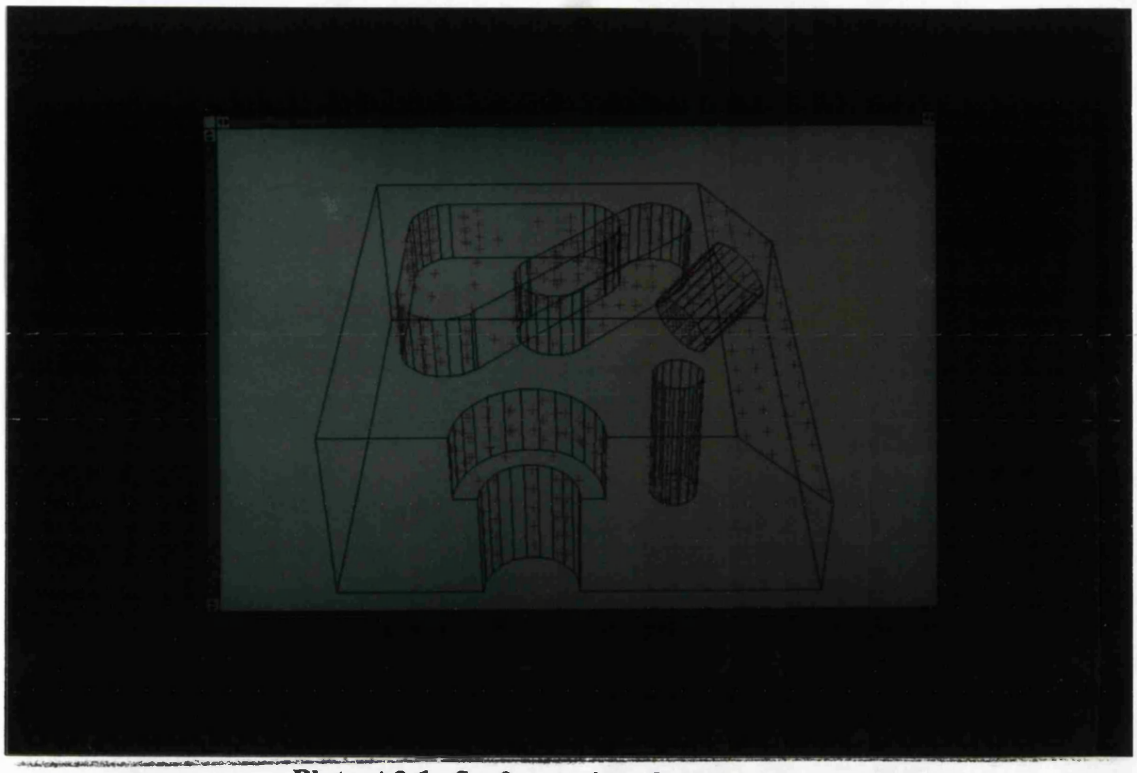


Plate A2.1 Surface points for test rig.

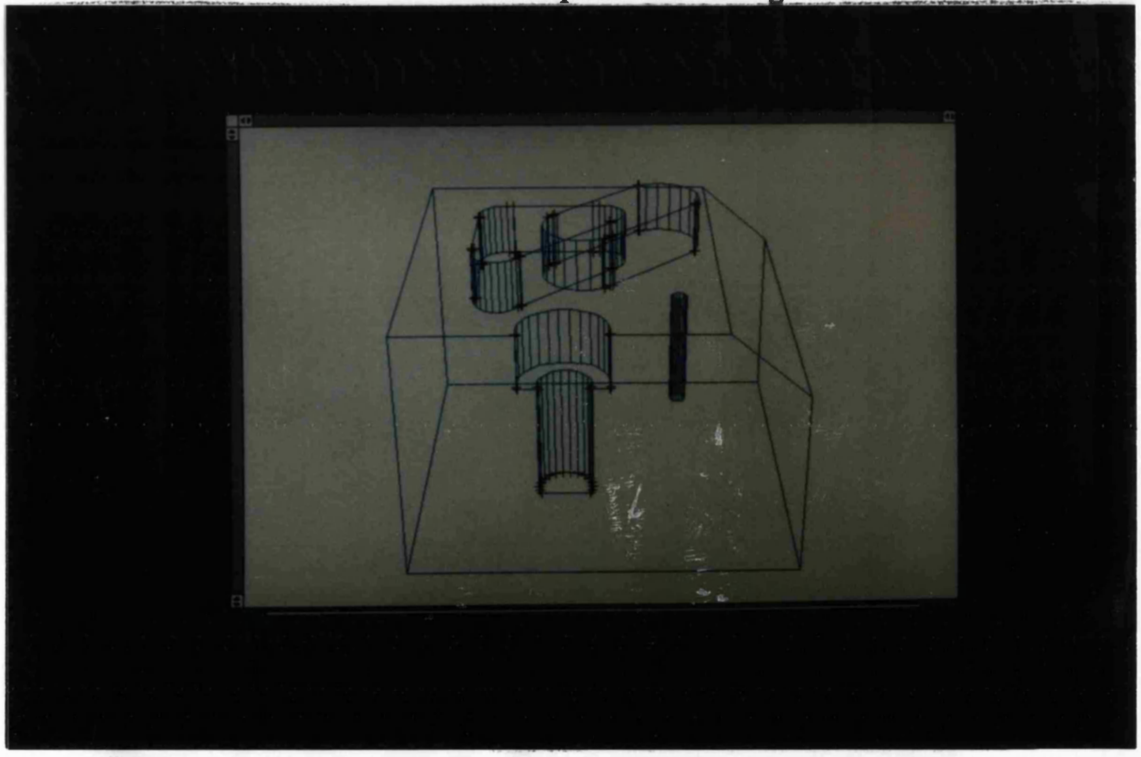


Plate A2.2 The topology changes which occur when test rig is grown for the probe arm. Crosses mark where new vertices occur.

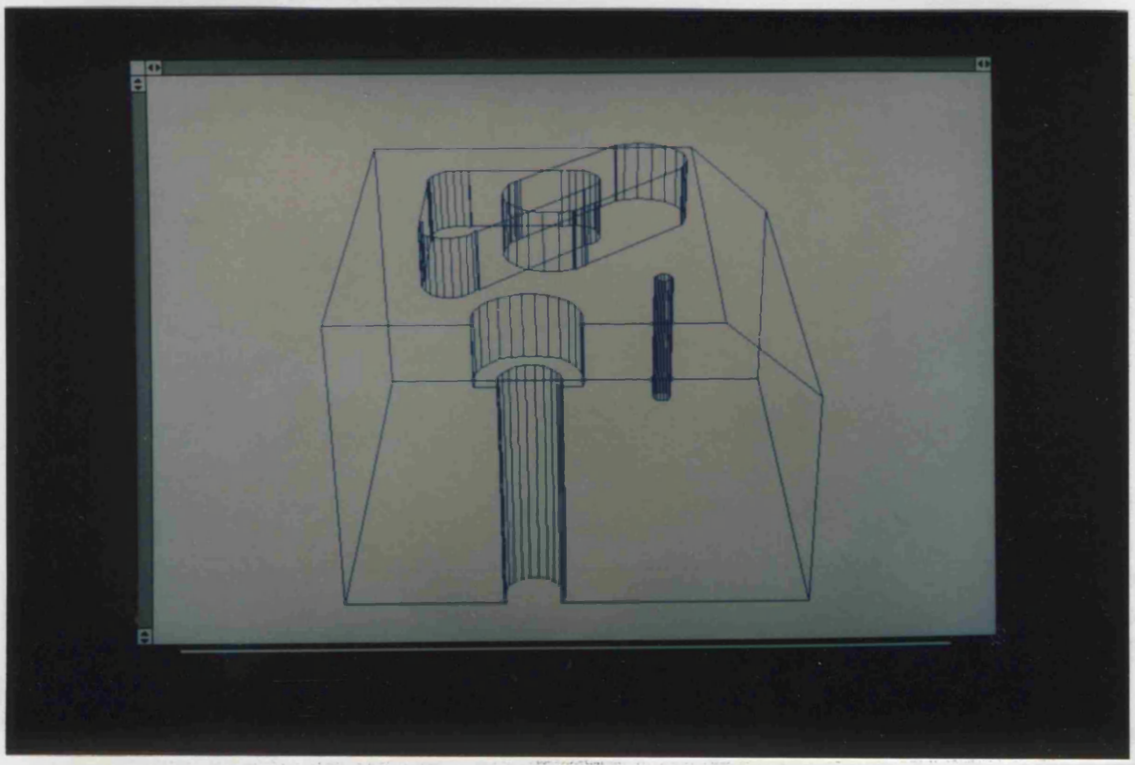


Plate A2.3 The grown test rig after correction.

Applications of 3-D Solid Modelling to Coordinate Measuring Inspection

I. Walker and A. F. Wallis

Manufacturing Group, School of Mechanical Engineering, Bath University,
Claverton Down, Bath, BA2 7AY

Summary

This paper describes a set of algorithms for the automation of component inspection using coordinate measuring machines. Solid modelling techniques are used to describe the component, and this model is then used to generate a set of inspection points and a collision-free probe path. The model is also used in the analysis of the resulting measurements. The algorithms have been implemented in a system that will handle the range of form, location and attitude tolerances, as defined by BS 308, for planar, cylindrical, conical and spherical features.

Introduction

Much research has been carried out into the automation and computerisation of various aspects of engineering design, component manufacture and inspection. One such area of research is the use of coordinate measuring machines (CMMs) for the automation of component inspection. The advantages of using such machines include a decrease in the time required for inspection, greater accuracy of inspection and greater flexibility.

Most CMMs are programmed manually by moving the measuring probe through a sequence of moves then it will later repeat. Developments in the computer software for use with CMMs has concentrated on simplifying this programming task and on the analysis of the measured data. Some work has been done on the automation of CMM programming, based on the generation of a database of paths for known features [1]. Other researchers have investigated the use of rule-based systems for the automatic generation of inspections procedures [2].

This paper presents a technique for the automatic inspection of components using a CMM, based on a computer solid model of the component.

The requirements of any technique that is to automate completely the use of CMMs for component inspection are:

1. A method of describing the shape of the component to be inspected, including its dimensions and tolerances.
2. An algorithm for the determining of a set of points at which to probe the surface of the component, based on the shape of its features and the specified tolerances.

3. An algorithm for generating a path for the CMM to follow such that the points may be probed.
4. A technique for relating the coordinates of the measured points to the component dimensions and tolerances, and reporting any that are in error.

This paper is divided into four main sections, based on each of these requirements.

Modelling the Component

Solid modelling is a technique for representing the shape of three-dimensional objects within a computer. It has been successfully used as the basis for automating a range of engineering tasks, including toolpath generation for NC machining and finite element mesh generation [3].

The details of the various schemes used to implement solid modelling, and the advantages and disadvantages of each have been described elsewhere [4]. The properties inherent in all solid modelling schemes that are of particular relevance to the work described in this paper are: solidity, rigidity and invariance under geometric transformations. Additional requirements for the modelling scheme are the ability to represent the range of surface geometries found in engineering components and the ability to support efficient and numerically robust algorithms.

The scheme chosen for this work is that of set-theoretic (CSG) solid modelling. In this scheme, an object is represented as the set-theoretic combination of simpler objects (*primitives*). The set-theoretic operators union, difference and intersection are used to build the required shape from these primitives. The primitives used are half-spaces - surfaces that divide three-dimensional space into regions that represent either solid or air. Using these primitives, a set of bounded shapes, such as cylinders, cuboids and spheres, can be constructed, and, using combinations of these and further half-spaces a wide range of engineering components may be modelled. Thus, although the surface of the model is made from parts of the surfaces of the various half-spaces, other parts of the half-space surfaces will lie inside or outside the model (in solid or air).

The solid modelling system used in this work allows the use of polynomial half-spaces, with the boundary being the zeros of a polynomial expression in the x, y and z coordinate variables. Most of the surfaces used in engineering components may thus be accurately modelled by the system.

Few solid modellering systems allow tolerance information to be included with the geometric information [5]. Clearly the representation of tolerances within the model is required for the automated generation of inspection points. The technique adopted is to allow the user to attach tolerance *attributes* to features on the model. This approach allows tolerances which apply to the geometry of shapes, i.e geometrical tolerances, to be defined as the model is defined. Geometrical tolerances have direct interpretation in terms of the geometric features of the object, and thus are a natural way of describing quality of shape.

The solid models used to represent components that are to be inspected are described using an input language [6] (graphical input is also available). The input language has a rich set of functions that may be used to define and manipulate shapes. Figure 1 shows part of the input file used to describe the component model shown in figure 2. The functions that are of particular importance to this work are the datum() and tol_set() functions. The first of these allows the user to define a surface as a datum for future reference. The second is used to attach a tolerance attribute to a feature. If this is a location or attitude tolerance, then the tolerance may relate to a datum surface defined previously. A general purpose attribute() function is used to define axes of features, select point generation grid sizes and set up coordinate datum surfaces for the CMM.

Generating the Probing Points

Having described the model of the component that is to be inspected, the next stage of the process is to generate a set of points that lie on the model corresponding to points on the component that are to be probed. These points must satisfy a number of constraints. Firstly they must lie on the surface of the model. Secondly the points must be reachable by the probe. Thirdly the set of points used to check each feature must be sufficient in number and in a pattern suitable for the inspection of that feature.

Points need only be generated on those model features that have been toleranced, or are datum surfaces. The number of points required for each feature is dependent on the tolerance applied to that feature (different tolerance types, in general, require a different number of points).

The method used to generate a set of surface points is first to generate a pattern of points on (parts of) the surface of the component model. A set of surface points is then chosen from this set. For the purpose of point and path generation and checking, a facetted approximation of the model is sufficient, so all non-planar surfaces in the model are facetted at this stage.

Each half-space in the model that represents part of a toleranced feature is considered separately. A regular rectangular grid, with a pitch that may be specified by the user when tolerancing the feature, is generated lying in the plane of the half-space. A point is positioned to lie within each grid square. These points are positioned at a (user-specified) random distribution of angle and direction from the centre of each square. This random offset reduces the likelyhood of regular or cyclic errors in the surface being undetected. As the points are generated they are tagged as belonging to a 'feature group'. Each group corresponds to a single feature in the model (which may comprise several half-spaces).

This set of points now needs to be processed to remove those points that do not lie on the surface of the model. Each point may be tested using a *membership test.*

This is a computationally simple task for a set-theoretic modelling system, and hence quick to perform. The point is compared with each half-space in the set-theoretic model and classified as to which side (air or solid) it is located. The half-space on which the point lies is treated as a special case and is classified as surface. These classifications may then be combined using DeMorgan's rules to yield a classification for the point. Those points that lie in 'air' or 'solid' are removed from further consideration.

Points that lie close to edges of the component, which may cause problems during inspection, may be detected by checking the distance from the point to the half-spaces that form edges in the component model.

Each surface point is now tested to see if the CMM probe can be positioned at that point without any part of the probe intersecting (colliding) with the component. The CMM used by the authors was a 3-axis machine with a touch probe fixed in a vertical orientation. For collision detection purposes, the probe may be modelled by two cylinders and a sphere as shown in figure 3. Thus the problem of detecting collisions is one of detecting intersections between the sphere and cylinders and the component model. In addition to checking for static collisions, collisions that occur when the probe is moving relative to the component need to be detected. One method of achieving this is to check for intersections between the volume swept by the probe model and the model of the component. Such null-object detection tests are time consuming, especially when the two models nearly, but do not quite, intersect.

The approach taken is to simplify the test by shrinking each of the elements in the probe model to a point separately, and creating new versions of the component model that are grown by a corresponding amount. Checking for static intersections between the probe model and the component model is then simplified to testing the point and the grown component model using the membership-test described previously. Furthermore, intersections that occur between the probe moving along a straight line path and the component may be found by detecting intersections between the straight-line corresponding to the path swept by the point and the grown models.

The task of detecting intersections between a line and a model is relatively simple to perform - it is the same as that required when generating ray-traced pictures of such models. The line is compared with each half-space in the model, and intersection points between the line and each half-space generated. Each of these points are membership-tested and if any of the tests result in a solid classification then the line intersects the model. The first, or last, point at which the line intersects the model may be found by sorting the points by their distance from one end of the line before testing each.

Figure 4 shows a version of the component model grown for the thin vertical cylinder of the probe. Note that the shapes of the various features in the model have changed (and one of the holes has disappeared completely). Details of the algorithm used are not given in this paper, but are given in [7]. It may however be noted that model growing, unlike scaling, is not a simple task.

A Z-safe plane is defined that is a horizontal plane positioned above the object to be measured. For each surface point, a probing point is generated that is offset by a small distance (typically a few millimeters) from the surface point in the direction of the coordinate axis that is closest to the outward-pointing surface normal vector direction. This is the location from which the CMM will probe the surface (the CMM is limited to probe in directions parallel to one of its coordinate axes). The surface normal vector is obtained directly from the half-space equation.

Owing to the fixed vertical attitude of the probe, all reachable points are approachable from above and so a default path to each surface point consisting of a vertical descent from the Z-safe plane to the probing point followed by a straight-line path to the surface point is assumed. The two line segments for this path are tested against each of the three grown models, and if any intersections are found, then point is rejected.

At this stage, a set of points has been generated, all of which lie of the surface of the model, and all of which may be reached by the probe. In general the original grid pitch is chosen to yield a number of points that is greater than will be required for the inspection of each feature. Points are now removed from the set in order to give a valid number of points for the inspection of each feature. The British Standard BS 7172 gives guidelines on selecting the number of points for checking a range of tolerance types, and on the distribution of such points, and these guidelines have been followed.

Surface points that are on the same feature are grouped into clusters of points that are closer together than the pitch of the point generation grid. For inspection purposes, only one point in each cluster will be selected, thus ensuring that the points chosen for measuring are widely-spaced. If more clusters remain than the number of points that are required, then the maximum size of each cluster is increased until the required number are left. Similarly, the cluster size is reduced if there are too few clusters. When the required number of clusters is reached, one point is selected from each cluster.

If insufficient points remain for any feature, then the user is warned. Further tests could be made at this stage to detect undesirable (for example co-linear) distributions of points.

The inspection points may be displayed on the computer screen, and points added or removed if desired. Figure 5 shows a set of surface points for checking some of the features in the model shown previously.

Generating the Probing Path

Having generated a set of surface and probing points, the next stage is to generate a path that, when followed by the CMM probe, will pass through all the points. In generating this path, there are three main aims:

1. The path must not result in any collisions between the probe and the component (assuming that the component is correct!).
2. The path should be such that the time taken to drive the probe along it is not excessive.
3. The time taken to generate the path should not be excessive.

This task is similar to the travelling salesman problem, which is well known to mathematicians. Many of the techniques used to solve that problem are not suitable for the probe path generation task due to the relatively large number of points and the lack of prior knowledge of the 'distance' between each point. An alternative heuristic approach is adopted.

Since the path from each probing point to its associated surface point has already been checked, the probe path is generated by constructing a path through the set of probing points, and then inserting probing moves at each of these points.

A cost function is defined which returns a value proportional to the time taken to drive the CMM probe between any two points. The current implementation calculates a cost dependent on the square root of the weighted addition of the distance between the points in each of the coordinate directions. The weighting values are used since the time taken for the CMM to move the same distance in each of the three coordinate directions is different. A more complicated cost function that takes account of the affect of accelerating and decelerating the probe CMM could be used.

The probe path is generated as a list of points that are either probing points or via-points. The probing-points are those generated previously, in addition the their (x,y,z) coordinates, are tagged with the probing direction vector, and a reference to the half-space on which the corresponding surface point lies. The via-points, which are inserted in order to avoid collisions between the probe and the component, are recorded as (x,y,z) coordinates.

The first probing point is chosen as the probing point with the smallest x-coordinate value. The next point is chosen by considering all of the probing points in the same feature group as the current point that have not yet been included in the path. The cost of moving to each of these points is calculated, and the point with the minimum cost is chosen as the next point and added to the probe path. This process is repeated until all the points in the current feature group have been added to the path. The next point is then chosen as the point (within any feature group) with the lowest cost and the current feature group updated. This process continues until all the points have been added to the path.

When calculating the cost of moving to the next probing point, the path to reach that point must be considered. Initially a straight-line path is assumed and this is tested for possible collisions using the method described in the previous section. If a collision is detected, then the path to this point must be modified. Two techniques are considered when modifying the path.

The simplest alternative, which is used if a better path cannot be found, is to insert two via-points at the height of the Z-safe plane, one directly above the current point and the other above the new probing point. This guarantees that the probe path will avoid any collision since a vertical path down to each probing point has already been checked, but clearly, large movements to and from the Z-safe plane may be generated.

A better path is searched for by modifying the direct straight-line path between the points. Starting at the points at which the path intersects the model (ie the points where the probe would hit the component), test vectors are generated that lie in the plane of the model surface at the collision point. The locations where these vectors leave the surface of the model are found, and the path modified to pass through a via-point that is offset from one of these points. The point chosen is the one closest to the destination point of the motion (see figure 6). This new path is then checked for collisions, and, if necessary, the process is repeated.

This concludes the path generation stage. The path may be displayed and could be amended at this stage if required. The probe path data is transferred to the CMM, which then follows the path, probing the component at each probing point. For each probing point, the coordinates where the probe contacts the component are recorded, together with its respective probing-point and the half-space and feature from which it was generated. An inspection path is shown in figure 7.

Analysing the Results

Each toleranced feature in the component is checked individually. The contact points that correspond to that feature are first extracted from the list of contact points. For some types of tolerance these points may be compared directly with the model. With other tolerance types, surfaces or axes have to be fitted to the measured points, and these compared to the tolerance zone defined by the tolerance specified for the feature.

Checking Planar Features

Flatness tolerances are checked by first fitting a plane to the set of measured points that correspond to a feature. The distance from each measured point to this best-fit plane is calculated and compared with the specified tolerance. Points that are out-of-tolerance are noted.

Attitude tolerances (squareness, parallelism and angularity) and position tolerances require comparison between the measured plane and some datum feature (a plane or line). If the datum plane is a measured (fitted) plane, then a transformation matrix is derived that maps the nominal datum plane onto the measured datum plane. This transformation is then applied to the nominal plane corresponding to the plane that is being checked. The distance from each measured point to this transformed plane is calculated, and points that are out-of-tolerance are noted.

If the datum feature is an axis (which cannot be measured directly), then this axis is first generated from the best-fit surface through the set of measured points for the datum feature (cylinder or cone).

Checking Non-planar Features

Roundness tolerances are checked by first fitting a surface (cylinder or sphere) to the set of measured points that correspond to a feature. The distance from each measured point to this best-fit surface is calculated and compared with the specified tolerance. Points that are out-of-tolerance are noted.

When checking attitude and position tolerances, a comparison has to be made between the position and orientation of a test-axis fitted to the measured points and a datum feature (a plane or axis). A tolerance zone, as defined in BS 308 Part 3, is generated about the nominal position of the feature being measured. The spatial limits of the zone are defined partly by attributes attached to the measured and datum features, and partly by the position of the measured feature's surfaces.

Test points are generated (using the line-model intersection test previously described) at locations where the test-axis intersects the surface of the nominal feature. These test points are compared with the tolerance zone, and, if they lie outside the zone, then the feature is out-of-tolerance.

The fitting of planes to point sets is achieved using principle components analysis. Other surfaces are fitted using an iterative algorithm developed at N.P.L. [8]. A report is printed that gives the deviation of each measured feature from its nominal position. Features that are out-of-tolerance are highlighted and the measured points corresponding to them may be displayed on the computer screen.

Implementation Details

The solid modelling techniques used by the software described in the paper are based on the work of Woodwark, Quinlan and Bowyer [6],[9]. An important aspect of the technique is that of spatial division and pruning. If the point membership tests used in the point and path generation and testing programs where implemented as described above, then the time taken to perform these tasks would be excessively large. In practice, the model is spatially divided into a number of simpler models, each of which is valid for a small region of space. When performing a membership-test, the point only need be compared with the simple model for the region that contains the point.

The software is written in a mixture of FORTRAN-77 and C, running on a Sun 3.

Conclusions

The work described in this paper has demonstrated that solid modelling techniques can be used to automate the planning and analysis stages of component inspection using CMMs. Set theoretic model definition has proved to be a natural base for attaching geometric tolerances to models. The point and path generation and measurement analysis algorithms have been tested on a range of models and have been found to be robust.

Acknowledgements

The project that this paper describes in part was supported by an SERC quota award.

References

[1] Raja, J. and Sheth, U.P. *Integration of Inspection into Automated Manufacturing Systems* Recent Developments in Production Research, pp. 119 - 124, Elsevier Science Publishers B.V., Amsterdam, 1988

[2] Park, H.D. and Mitchell, O.R. *CAD based Planning and Execution of Inspection* Proc. Computer Society Conference of Computer Vision and Pattern Recognition, pp. 858 - 863, 1988

[3] Woodwark, J.R. *Shape models in Computer Integrated Manufacture - a review* CAE Journal, June 1988, pp. 103 - 112

[4] Requicha, A.A.G. *Representations for Rigid Solids: Theory, Methods and Systems* Computing Surveys, Vol. 12, No. 4, pp. 437 - 461, Dec. 1980

[5] Requicha, A.A.G. *Representation of tolerances in solid modelling: issues and alternative approaches* Proc. General Motors Symposium, Detroit, USA, 1983, pp. 3 - 22

[6] Woodwark, J.R., and Bowyer, A. *Better and Faster Pictures from Solid Models* IEE Computer Aided Engineering Journal, V3, No 2 (1986)

[7] Walker, I. *CMM Path Planning and Solid Modelling*, PhD. thesis (in preparation), University of Bath.

[8] Forbes, A.B., *Least Squares Best-fit Geometric Elements*, Internal NPL Report DITC 140/89, April 1989.

[9] Woodwark, J.R. and Quinlan, K.M. *Reducing the Effect of Complexity on Volume Model Evaluation* CAD Journal, Vol. 14, 1982, pp. 89 - 95.

```
FUNCTION create_top_block (in_dev:Real): Set

Integers {num_fac}
Reals {r_prop}
Sets {end_cyls,mid_box,res_pro,sid_spac}

{
  block_end_pt := pt(45.0,35.0,25.0)

  r_prop := 8.0/in_dev
  num_fac := cylfacetnum (8.0,r_prop)

  end_cyls := cylinder( ln (z_dir,blovk_end_pt ) ,8.0,num_fac)
              & space (z_dir,pt (45.0,45.0,54.0))

  end_cyls := attribute (end_cyls,gen_feat,
       "CYL_AXIS : D0.0 0.0 1.0 R8.0 P45.0 35.0 25.0");

  end_cyls := tol_set (end_cyls,PARALLELISM,0.1,pt(1.0,0.0,0.0),"-DZ0_AX")

  res_pro := slide (end_cyls,ln(x_dir,blovk_end_pt),30.0)
  res_pro := spin (res_pro,ln(z_dir,blovk_end_pt),PI/4.0)
  res_pro := attribute (res_pro,gen_feat,
       "CYL_AXIS : D0.0 0.0 1.0 R8.0 P66.2132 56.2132 25.0");

  res_pro := tol_set (res_pro,PARALLELISM,0.1,pt(0.0,0.0,0.0),"-DZ0_AX")
  res_pro := tol_set (res_pro,SQUARENESS,0.1,pt(0.0,1.0,0.0),"-DZ1_DTM")

  res_pro := end_cyls | res_pro

  mid_box := space(-x_dir,blovk_end_pt)& space (x_dir,pt(75.0,35.0,25.0))
    & space (z_dir,pt(45.0,45.0,54.0)) & space (-z_dir,pt(45.0,35.0,35.0))
  mid_box := spin (mid_box,ln(z_dir,blovk_end_pt),PI/4.0)

  sid_spac := space (-y_dir,pt(45.0,27.0,25.0))
  sid_spac := spin (sid_spac,ln(z_dir,blovk_end_pt),PI/4.0)
  sid_spac := tol_set (sid_spac,PARALLELISM,0.1,pt(0.0,0.0,0.0),"-DYANG_FACE")
  mid_box := mid_box & sid_spac

  sid_spac := space(y_dir,pt(45.0,43.0,25.0))
  sid_spac := spin (sid_spac,ln(z_dir,blovk_end_pt),PI/4.0)
  sid_spac := datum (sid_spac,"YANG_FACE")
  mid_box := mid_box & sid_spac

  res_pro := res_pro | mid_box

  RETURN (res_pro)
}
```

Figure 1. Model definition

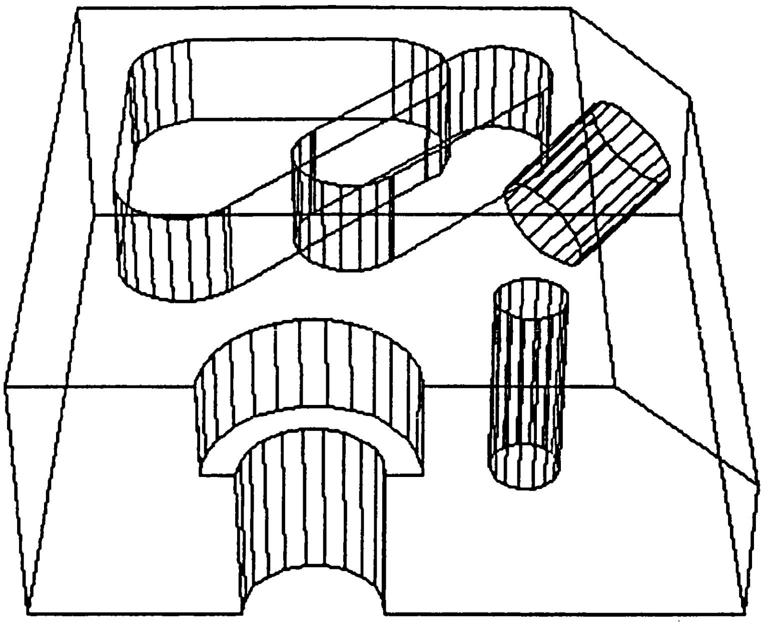


Figure 2. An image of the component model

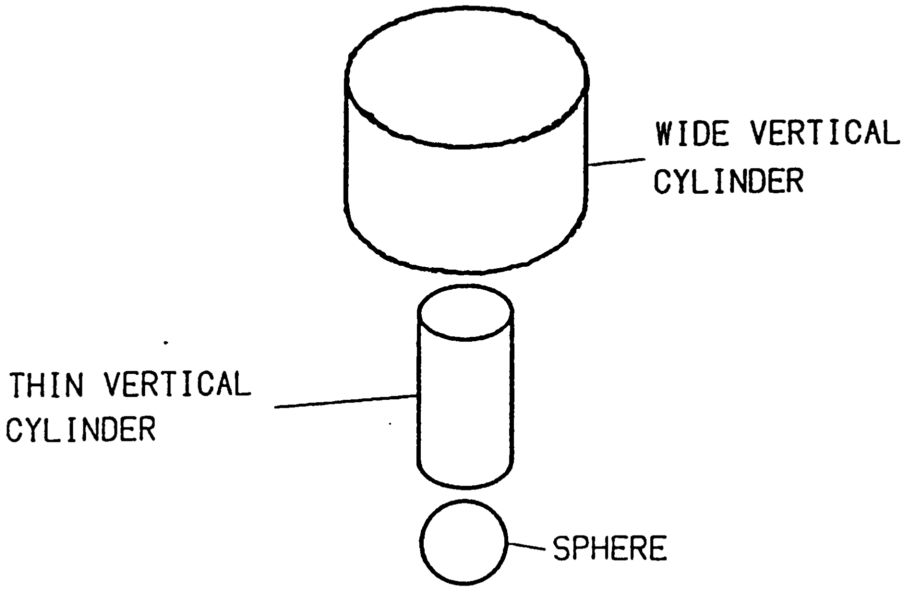


Figure 3. The probe model

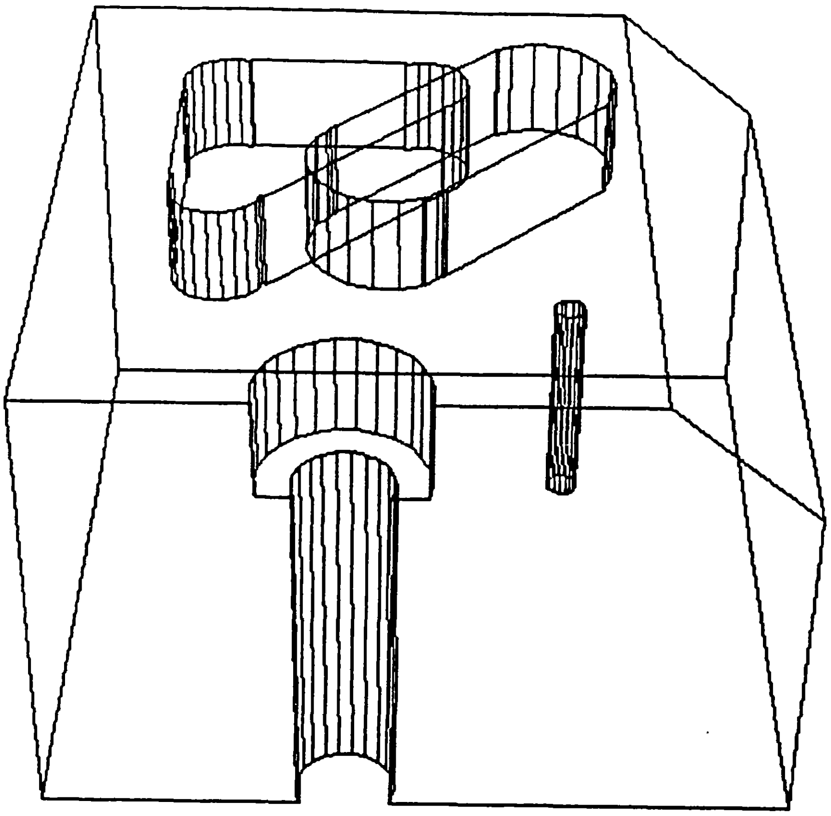


Figure 4. A 'grown' component model

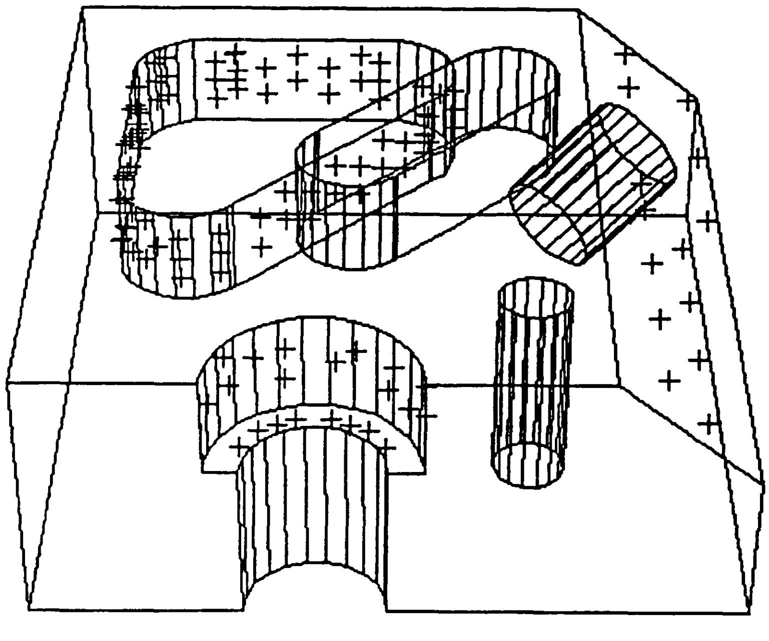


Figure 5. A set of surface points

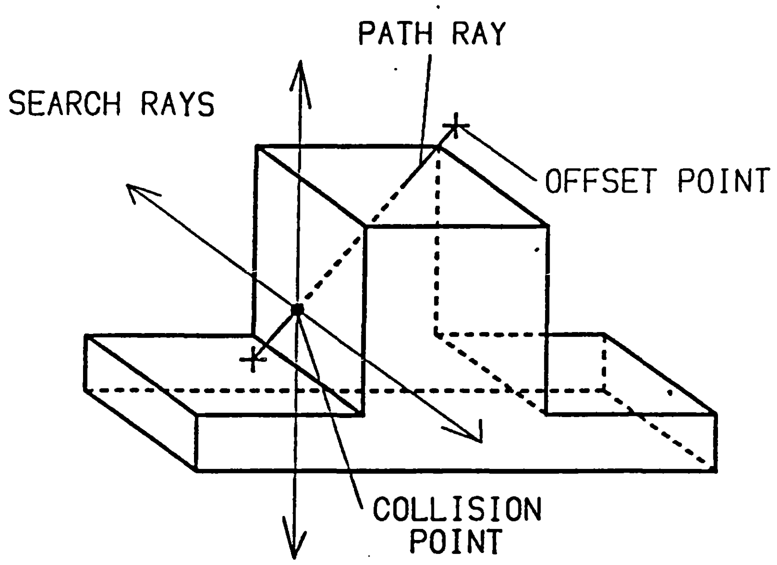


Figure 6a. Generating test vectors from a collision point

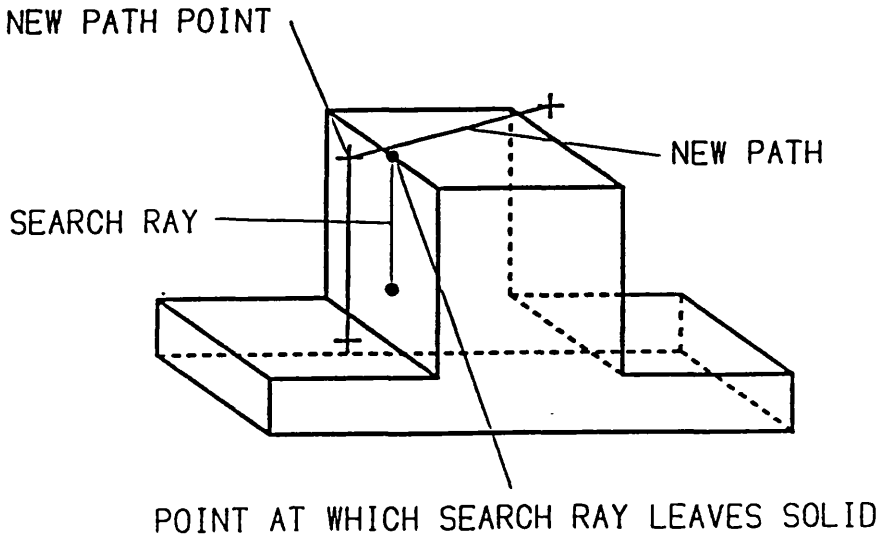


Figure 6b. A modified path segment with new path (via) point

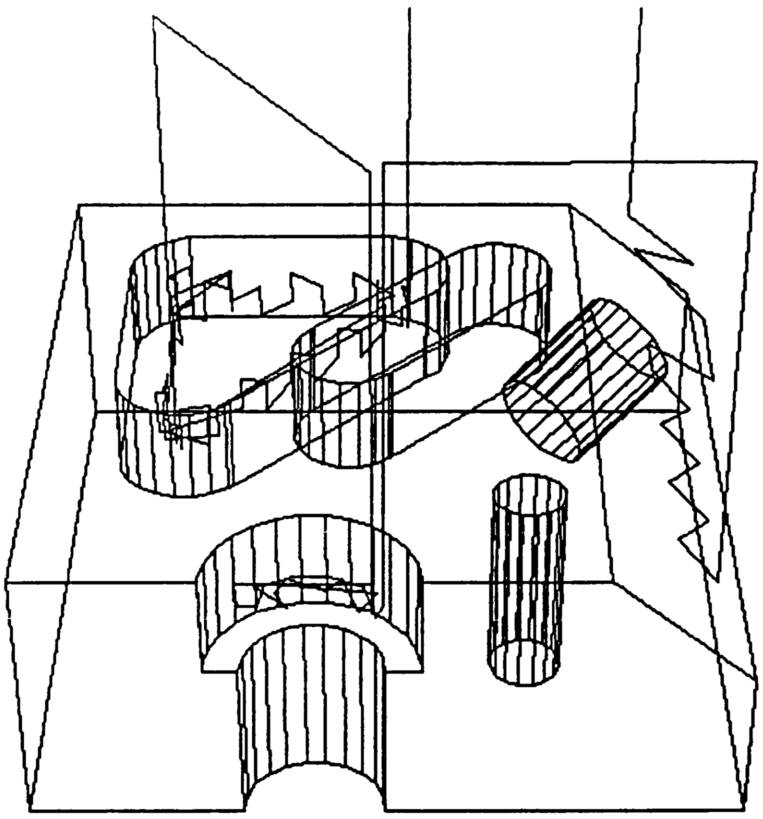


Figure 7. A probe path